Introduction to Software Architecture and Documentation

Claire Le Goues

Michael Hilton

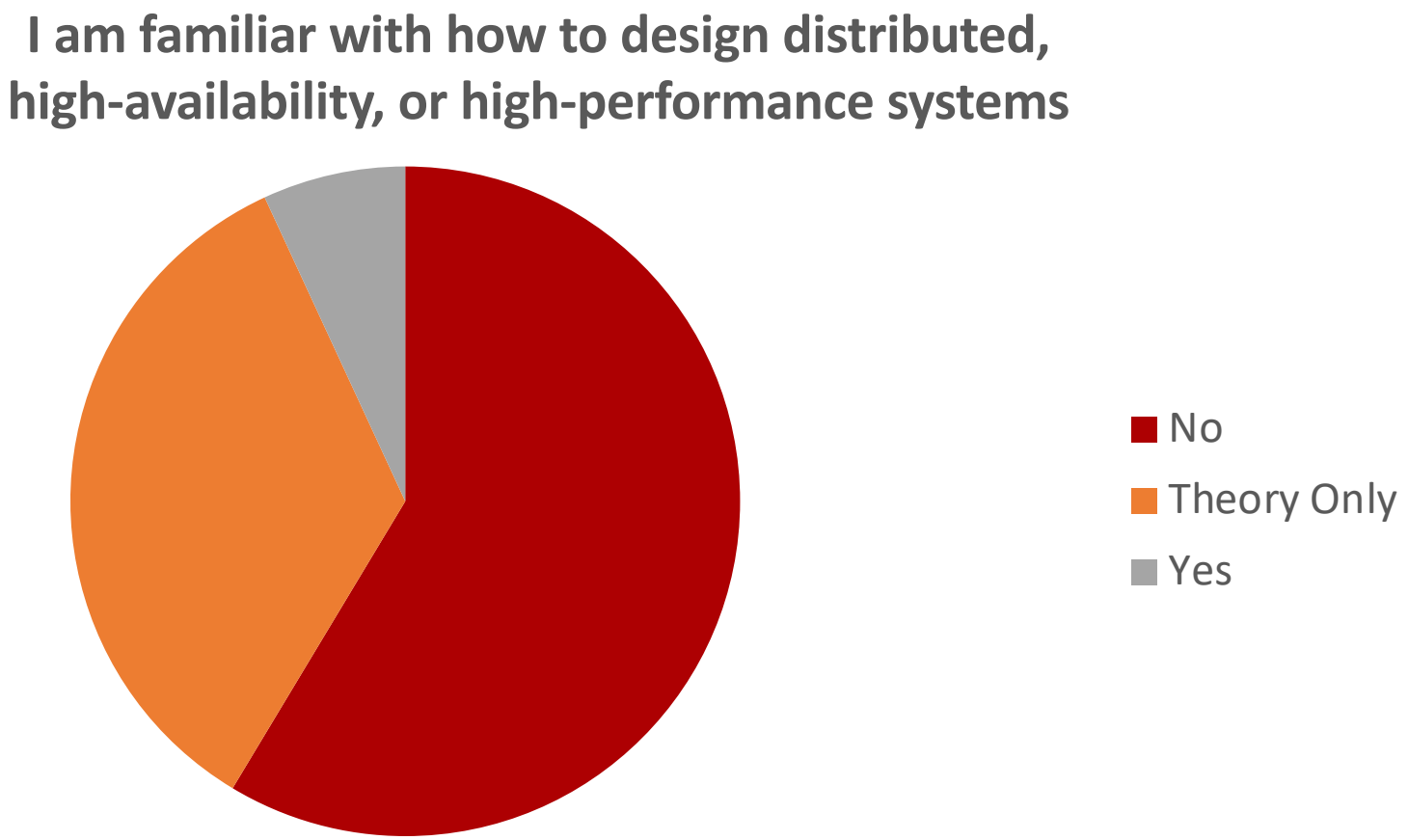
Administrivia

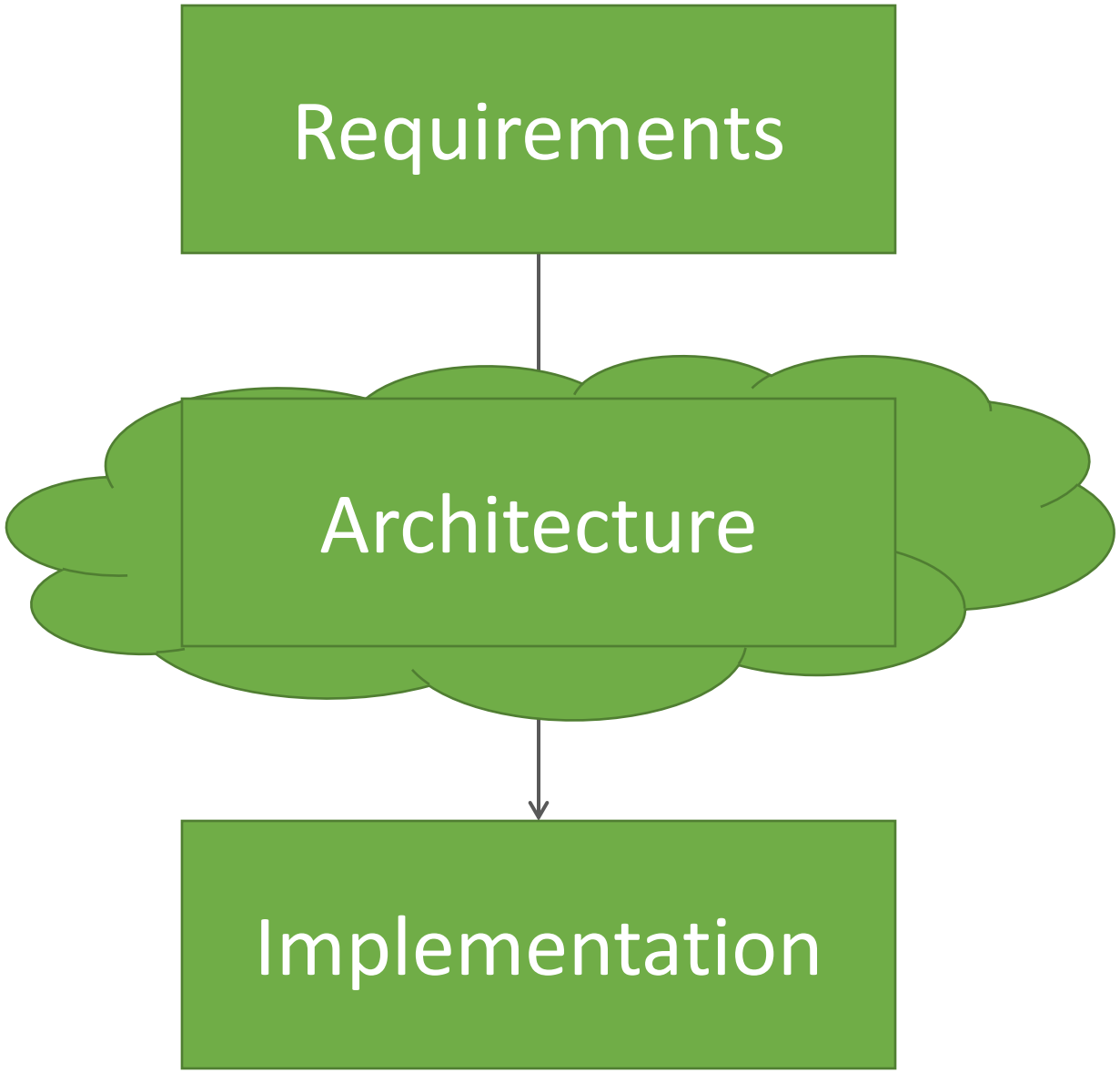
- Recitation instructions posted: do some install ahead of time.
- Schedule your HW3 interviews!

Learning Goals

- Understand the abstraction level of architectural reasoning
- Approach software architecture with quality attributes in mind
- Distinguish software architecture from (object-oriented) software design
- Use notation and views to describe the architecture suitable to the purpose
- Document architectures clearly, without ambiguity

About You


I am familiar with how to design distributed, high-availability, or high-performance systems
No
Theory Only
Yes


Requirements
Architecture
Implementation

Quality Requirements, now what?

- "should be highly available"
- "should answer quickly, accuracy is less relevant"
- "needs to be extensible"
- "should efficiently use hardware resources"

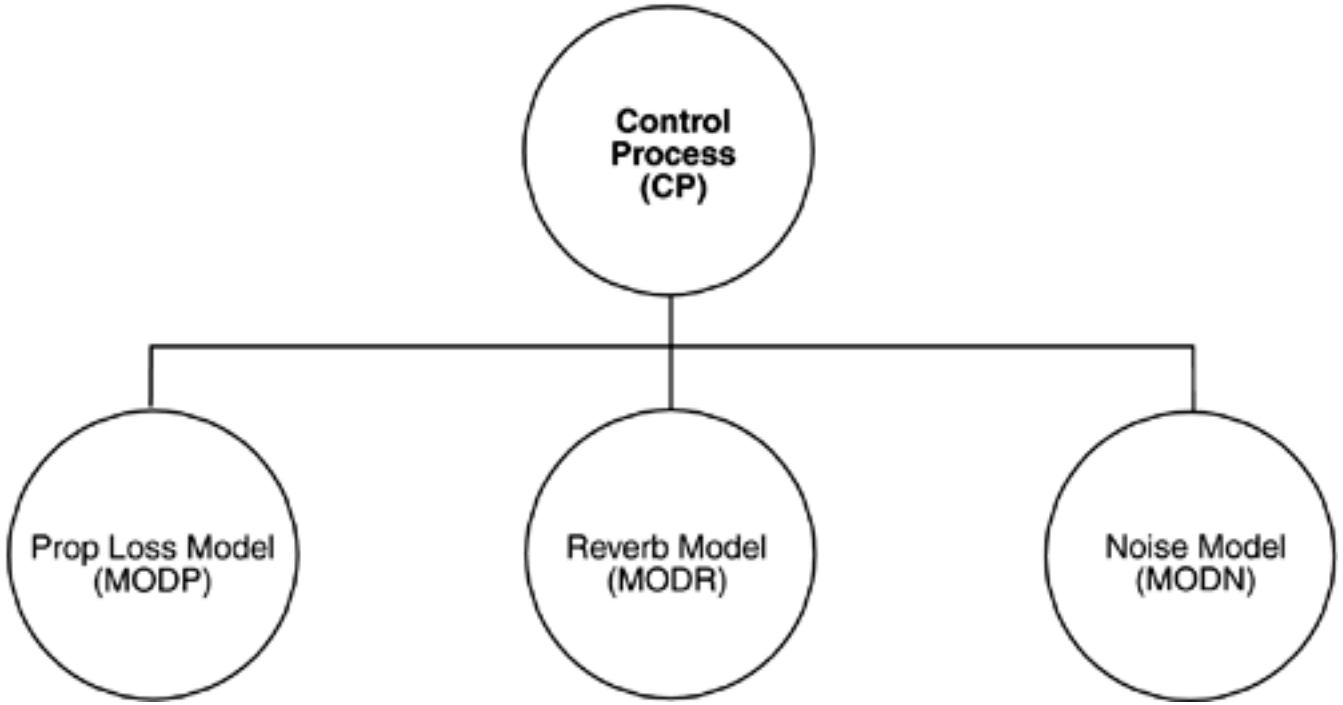
SOFTWARE ARCHITECTURE

Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

[Bass et al. 2003]

Note: this definition is ambivalent to whether the architecture is known, or whether it's any good!


Control Process (CP)
Prop Loss Model (MODP)
Reverb Model (MODR)
Noise Model (MODN)

Why Architecture? [BCK03]

- Represents earliest design decisions.
- Aids in communication with stakeholders
  - Shows them “how” at a level they can understand, raising questions about whether it meets their needs
- Defines constraints on implementation
  - Design decisions form “load-bearing walls” of application
- Dictates organizational structure
  - Teams work on different components
- Inhibits or enables quality attributes
  - Similar to design patterns
- Supports predicting cost, quality, and schedule
  - Typically by predicting information for each component
- Aids in software evolution
  - Reason about cost, design, and effect of changes
- Aids in prototyping
  - Can implement architectural skeleton early

Beyond functional correctness

- Quality matters, eg.,
  - Performance
  - Availability
  - Modifiability, portability
  - Scalability
  - Security
  - Testability
  - Usability
  - Cost to build, cost to operate

CASE STUDY: ARCHITECTURE AND QUALITY AT TWITTER

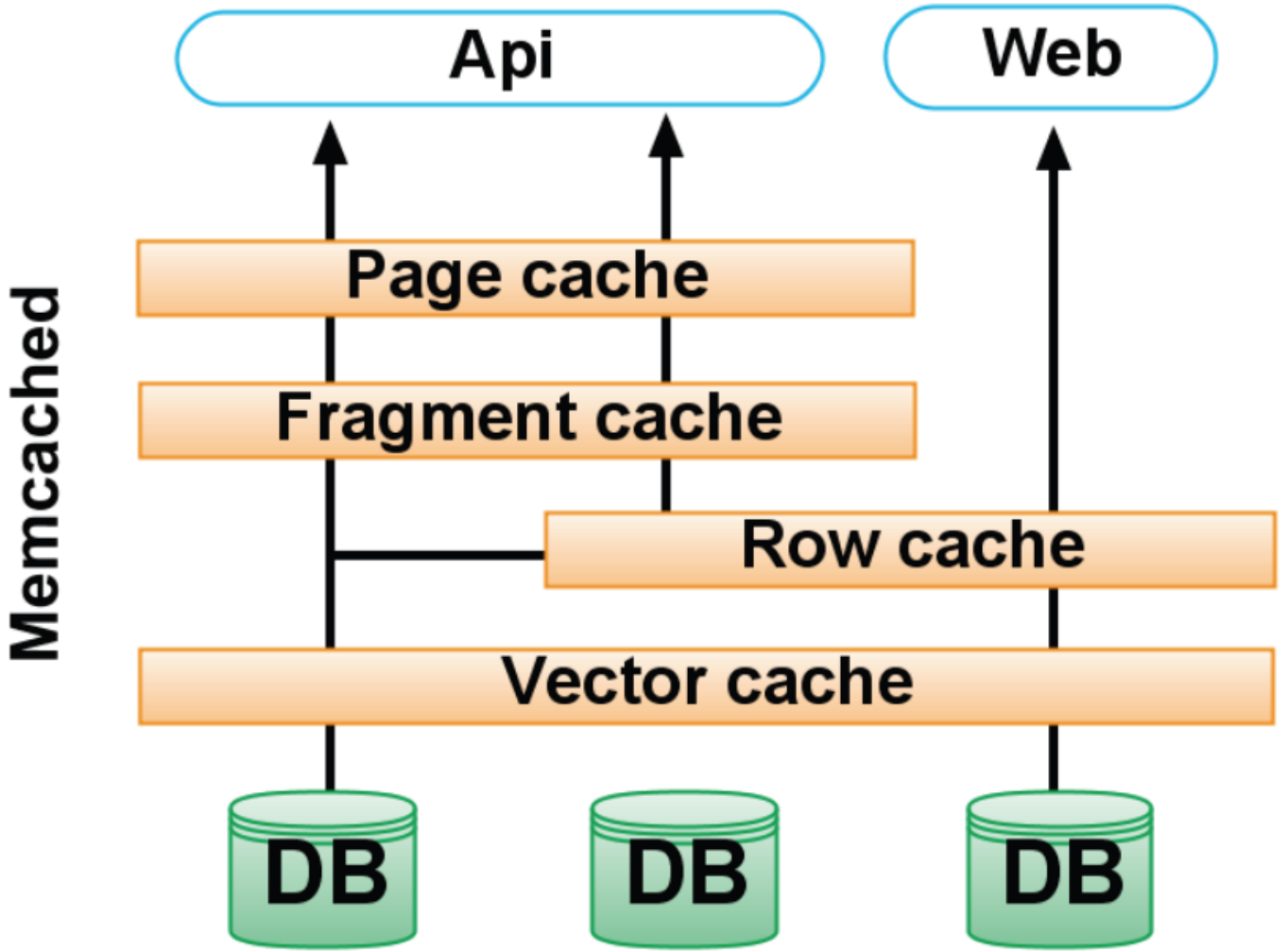

twitter
Home Public Timeline Help
Twitter is over capacity.
Too many tweets! Please wait a moment and try again.
© 2009 Twitter About Us Contact Blog Status API Help Jobs TOS Privacy


SOUTH
AFRICA
2010
FIFA
WORLD CUP

Inspecting the State of Engineering

- Running one of the world's largest Ruby on Rails installations
  - 200 engineers
- Monolithic: managing raw database, memcache, rendering the site, and presenting the public APIs in one codebase
- Increasingly difficult to understand system; organizationally challenging to manage and parallelize engineering teams
- Reached the limit of throughput on our storage systems (MySQL); read and write hot spots throughout our databases
- Throwing machines at the problem; low throughput per machine (CPU + RAM limit, network not saturated)
- Optimization corner: trading off code readability vs performance

Caching


Api
Web
Memcached
Page cache
Fragment cache
Row cache
Vector cache
DB
DB
DB

Twitter's Quality Requirements/Redesign goals??

- Improve median latency; lower outliers
- Reduce number of machines 10x
- Isolate failures
- "We wanted cleaner boundaries with “related” logic being in one place"
  - encapsulation and modularity at the systems level (rather than at the class, module, or package level)
- Quicker release of new features
  - "run small and empowered engineering teams that could make local decisions and ship user-facing changes, independent of other teams"

performance

reliability

maintainability

modifiability

JVM vs Ruby VM

- Rails servers capabile of 200-300 requests / sec / host
- Experience with Scala on the JVM; level of trust
- Rewrite for JVM allowed 10-20k requests / sec / host

Programming Model

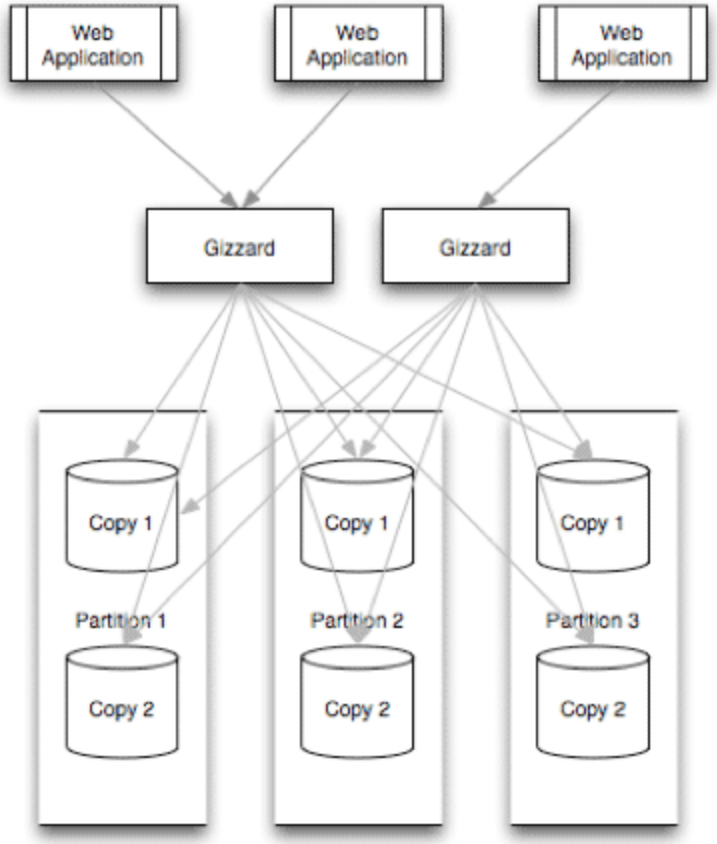
- Ruby model: Concurrency at process level; request queued to be handled by one process
- Twitter response aggregated from several services – additive response times
- "*As we started to decompose the system into services, each team took slightly different approaches. For example, the failure semantics from clients to services didn't interact well: we had no consistent back-pressure mechanism for servers to signal back to clients and we experienced "thundering herds" from clients aggressively retrying latent services.*"
- Goal: Single and uniform way of thinking about concurrency
  - Implemented in a library for RPC (Finagle), connection pooling, failover strategies and load balancing

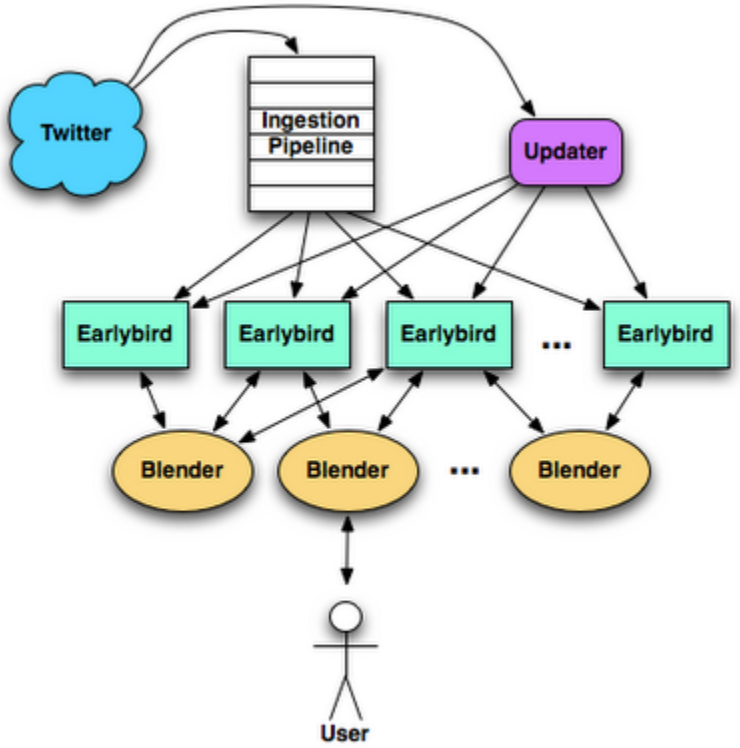
Independent Systems

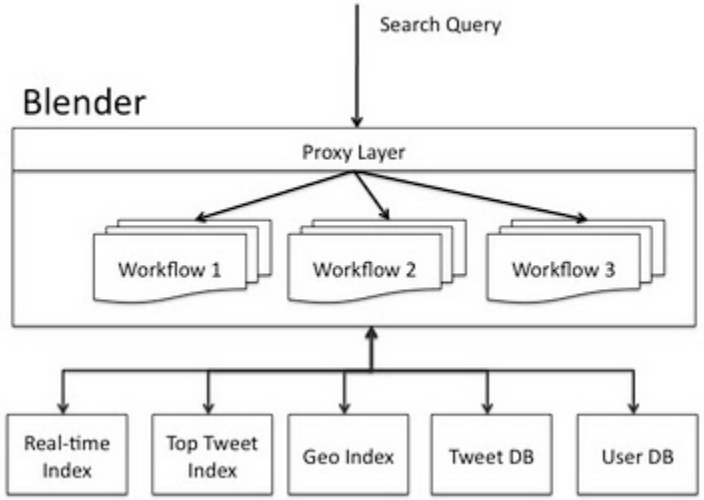
- "In our monolithic world, we either needed experts who understood the entire codebase or clear owners at the module or class level. Sadly, the codebase was getting too large to have global experts and, in practice, having clear owners at the module or class level wasn't working. Our codebase was becoming harder to maintain, and teams constantly spent time going on "archeology digs" to understand certain functionality. Or we'd organize "whale hunting expeditions" to try to understand large scale failures that occurred."
- From monolithic system to multiple services
  - Agree on RPC interfaces, develop system internals independently
  - Self-contained teams

Storage

- Single-master MySQL database bottleneck despite more modular code
- Temporal clustering
  - Short-term solution
  - Skewed load balance
  - One machine + replications every 3 weeks
- Move to distributed database (Glizzard on MySQL) with "roughly sortable" ids
- Stability over features – using older MySQL version



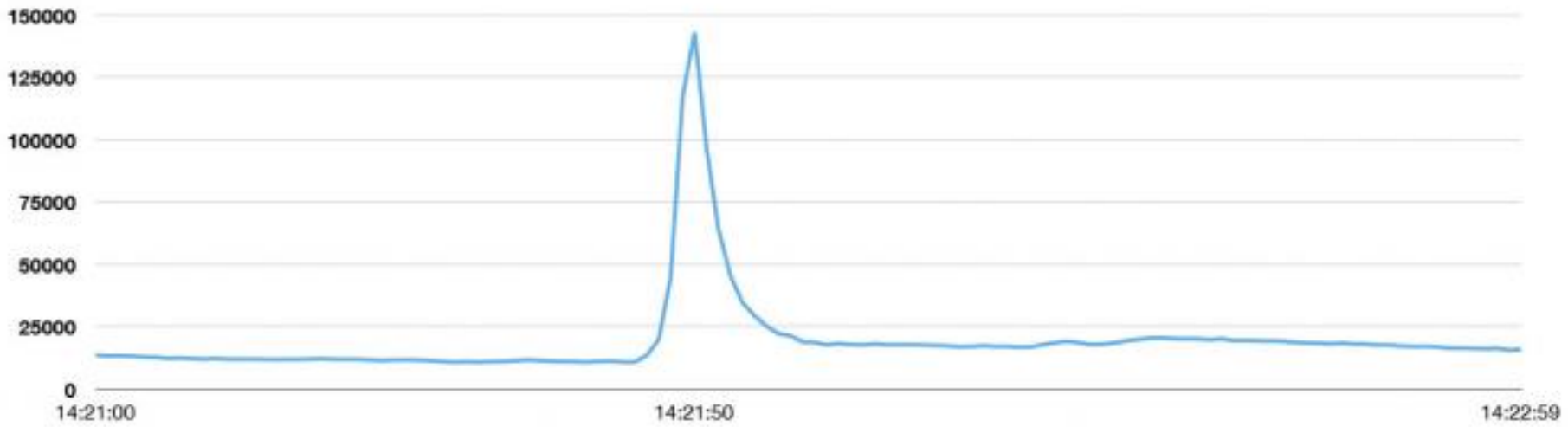

Twitter
Ingestion Pipeline
Updater
Earlybird
Earlybird
Earlybird
...
Earlybird
Blender
Blender
...
Blender
User


Search Query
Blender
Proxy Layer
Workflow 1
Workflow 2
Workflow 3
Real-time Index
Top Tweet Index
Geo Index
Tweet DB
User DB

Data-Driven Decisions

- Many small independent services, number growing
- Own dynamic analysis tool on top of RPC framework
- Framework to configure large numbers of machines
  - Including facility to expose feature to parts of users only


天空の城
ラピュタ



On Saturday, August 3 in Japan, people watched an airing of Castle in the Sky, and at one moment they took to Twitter so much that we hit a one-second peak of 143,199 Tweets per second.

Key Insights: Twitter Case Study

- Architectural decisions affect entire systems, not only individual modules
- Abstract, different abstractions for different scenarios
- Reason about quality attributes early
- Make architectural decisions explicit
- Question: *Did the original architect make poor decisions?*

ARCHITECTURE VS OBJECT-LEVEL DESIGN

Levels of Abstraction

- Requirements
  - high-level “what” needs to be done
- Architecture (High-level design)
  - high-level “how”, mid-level “what”
- OO-Design (Low-level design, e.g. design patterns)
  - mid-level “how”, low-level “what”
- Code
  - low-level “how”

Design vs. Architecture

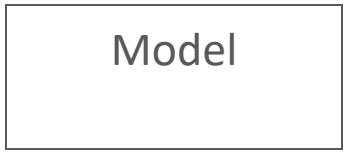
Design Questions

- How do I add a menu item in Eclipse?
- How can I make it easy to add menu items in Eclipse?
- What lock protects this data?
- How does Google rank pages?
- What encoder should I use for secure communication?
- What is the interface between objects?

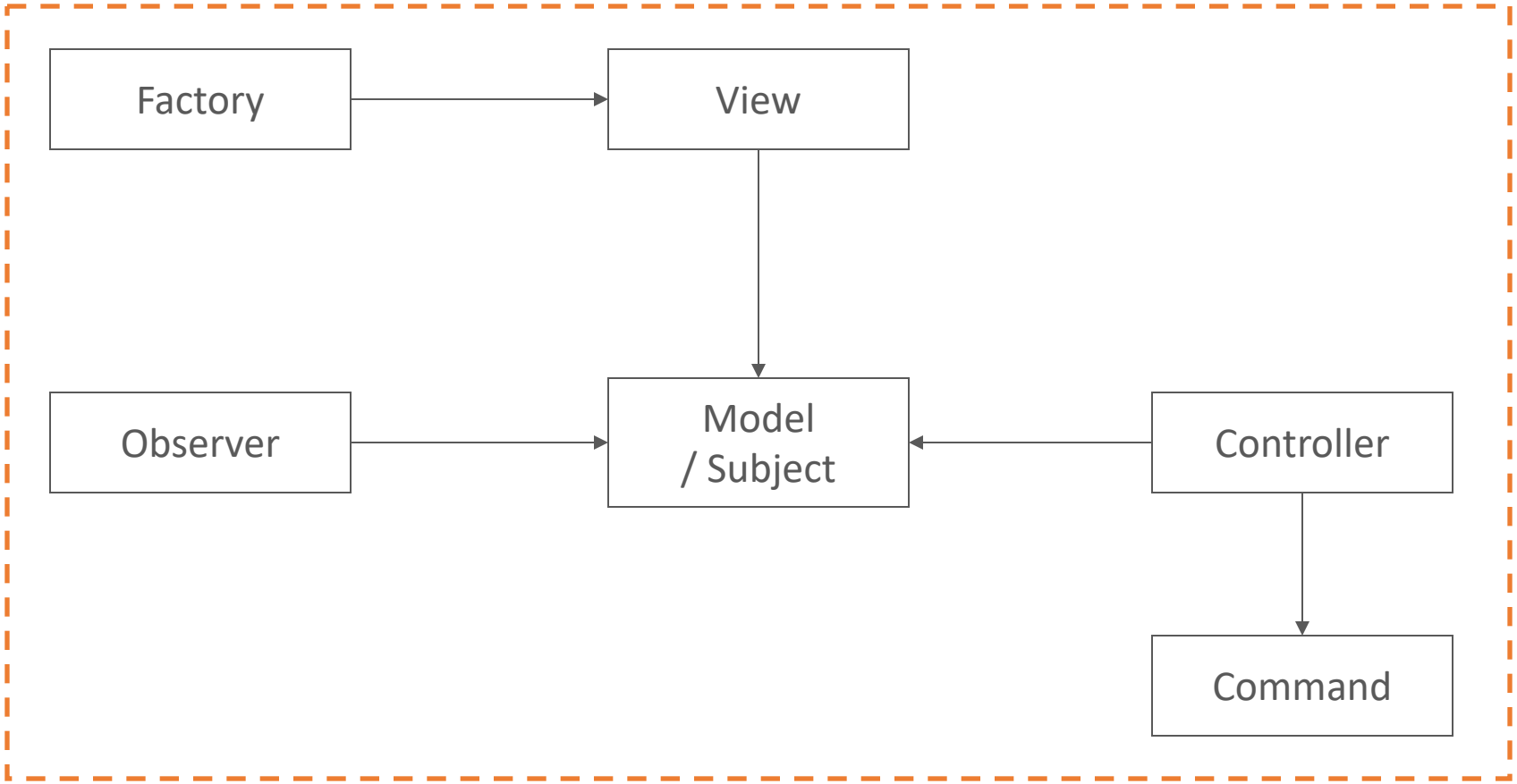
Architectural Questions

- How do I extend Eclipse with a plugin?
- What threads exist and how do they coordinate?
- How does Google scale to billions of hits per day?
- Where should I put my firewalls?
- What is the interface between subsystems?

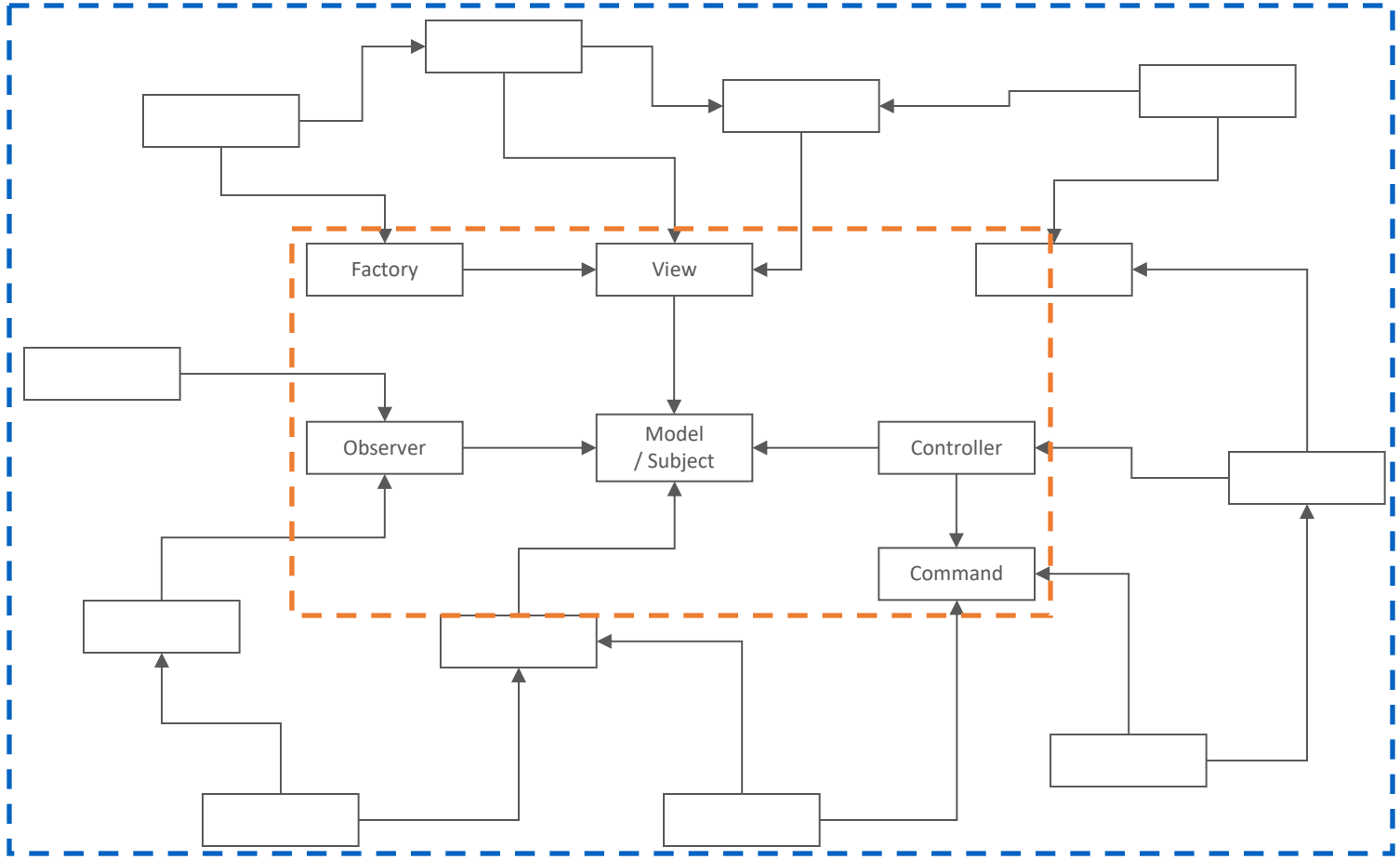
Objects



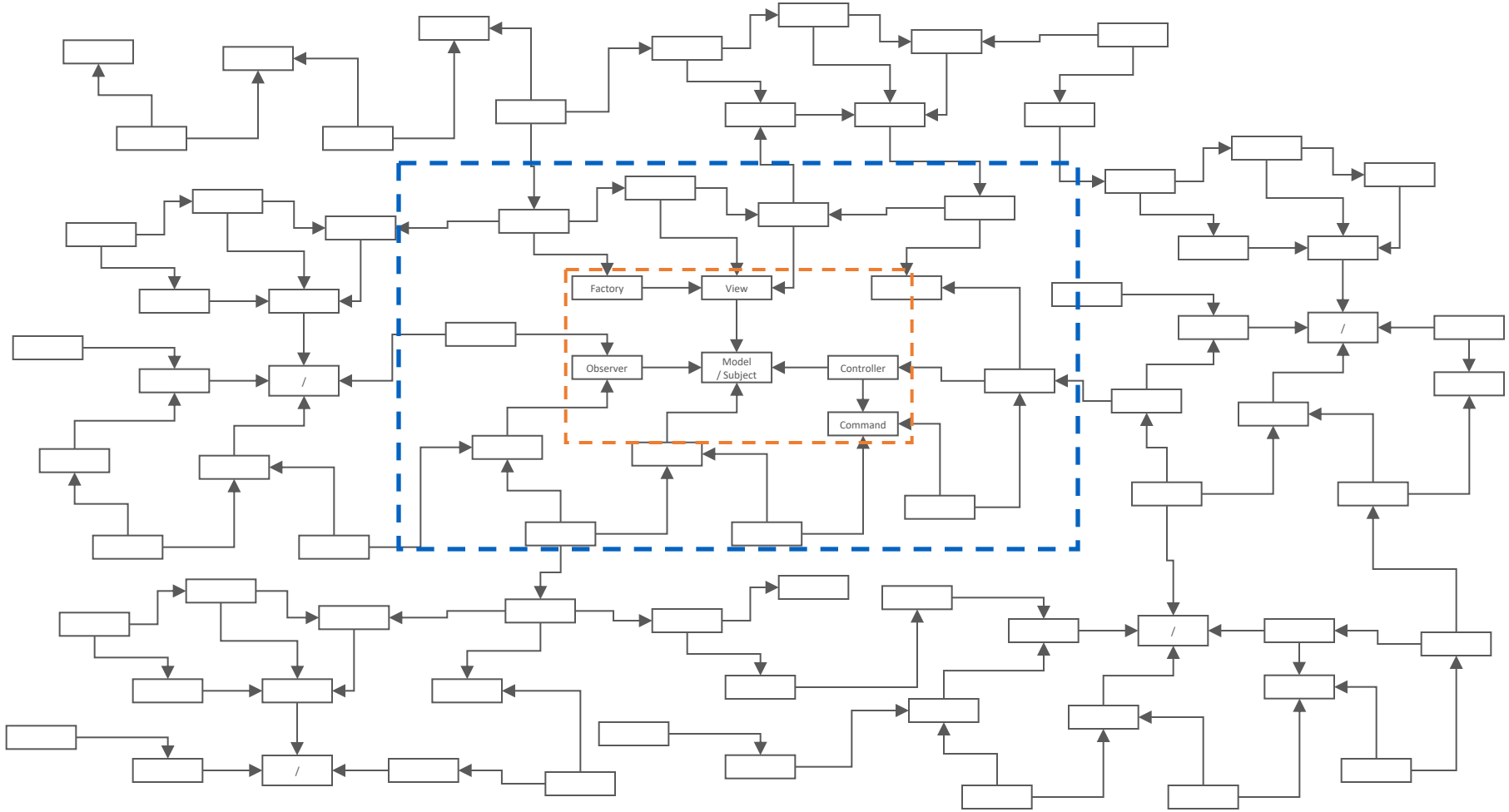
Design Patterns



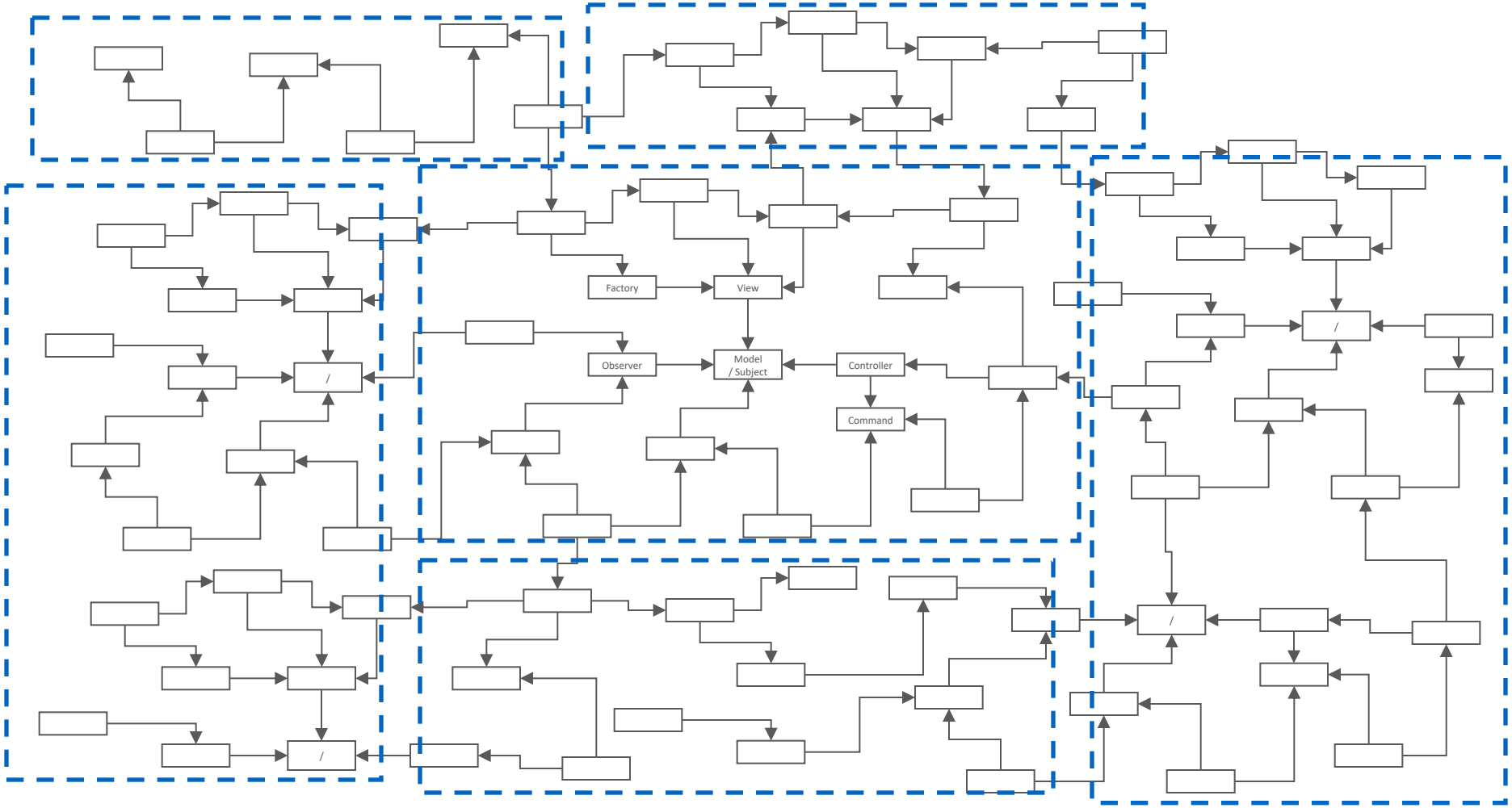
Design Patterns



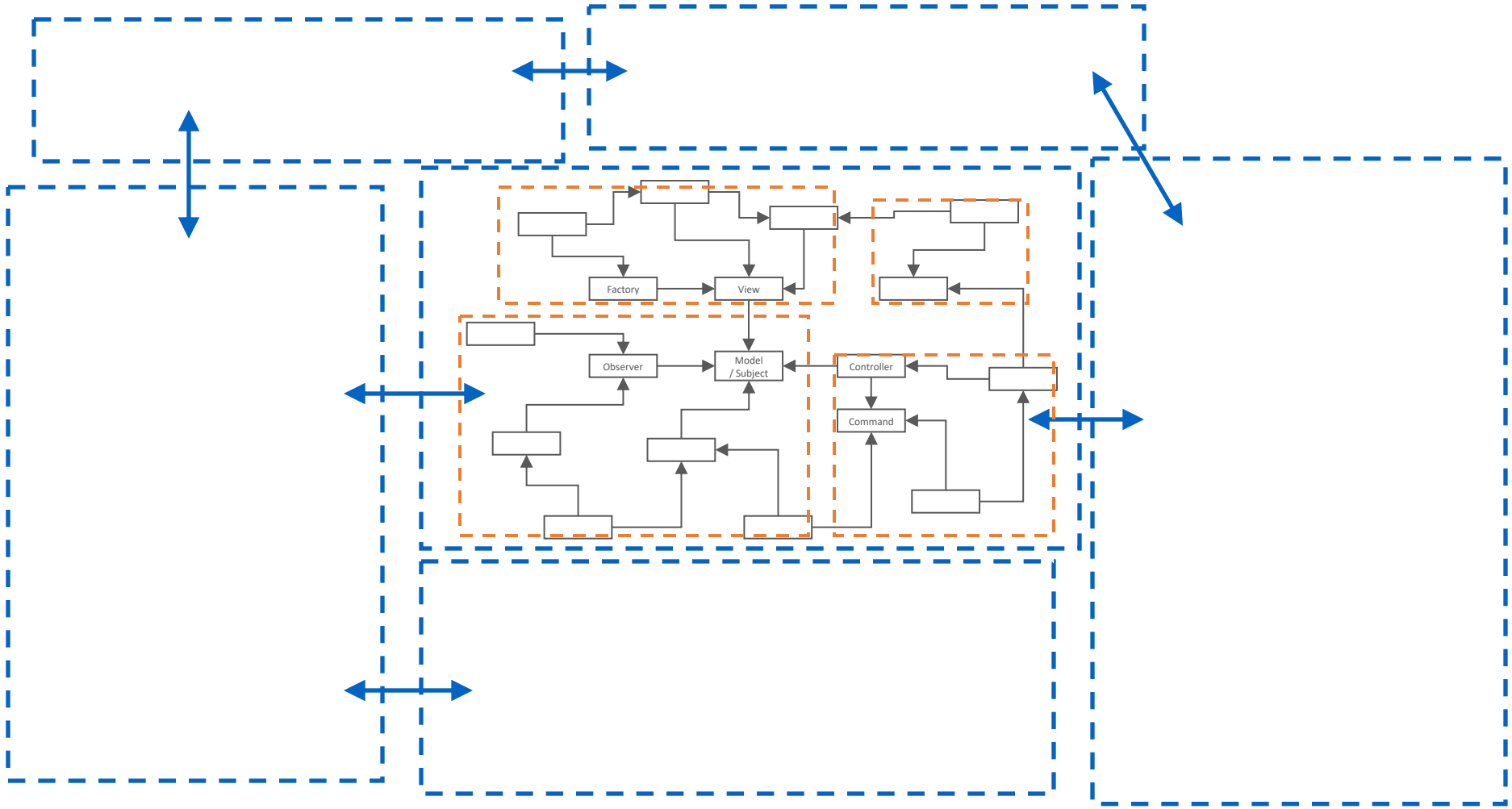
Design Patterns



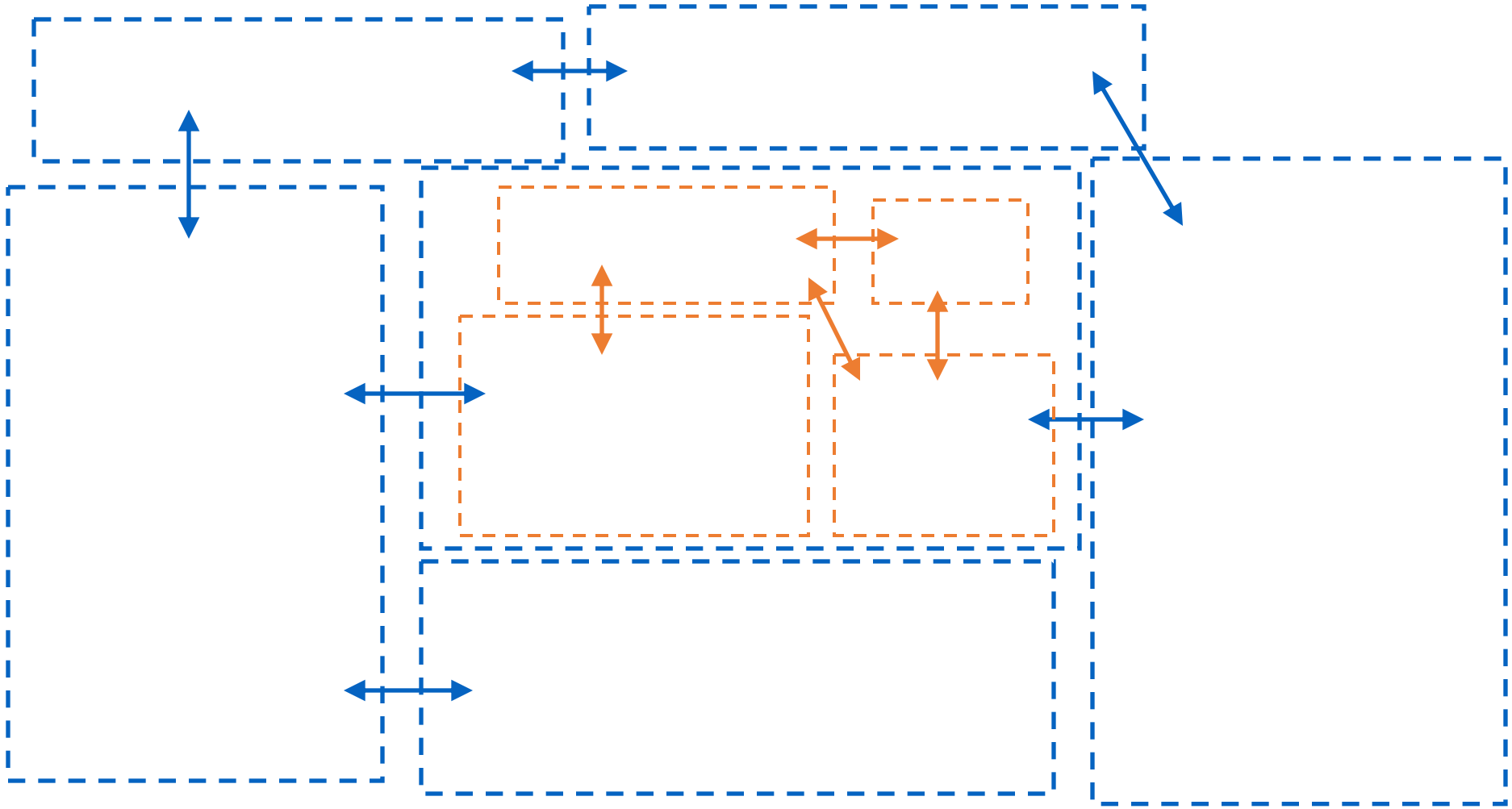
Architecture



Architecture



Architecture

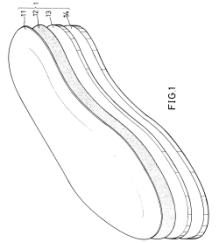


ARCHITECTURE DOCUMENTATION & VIEWS

Architecture Disentangled

Architecture as
structures and relations
(the actual system)

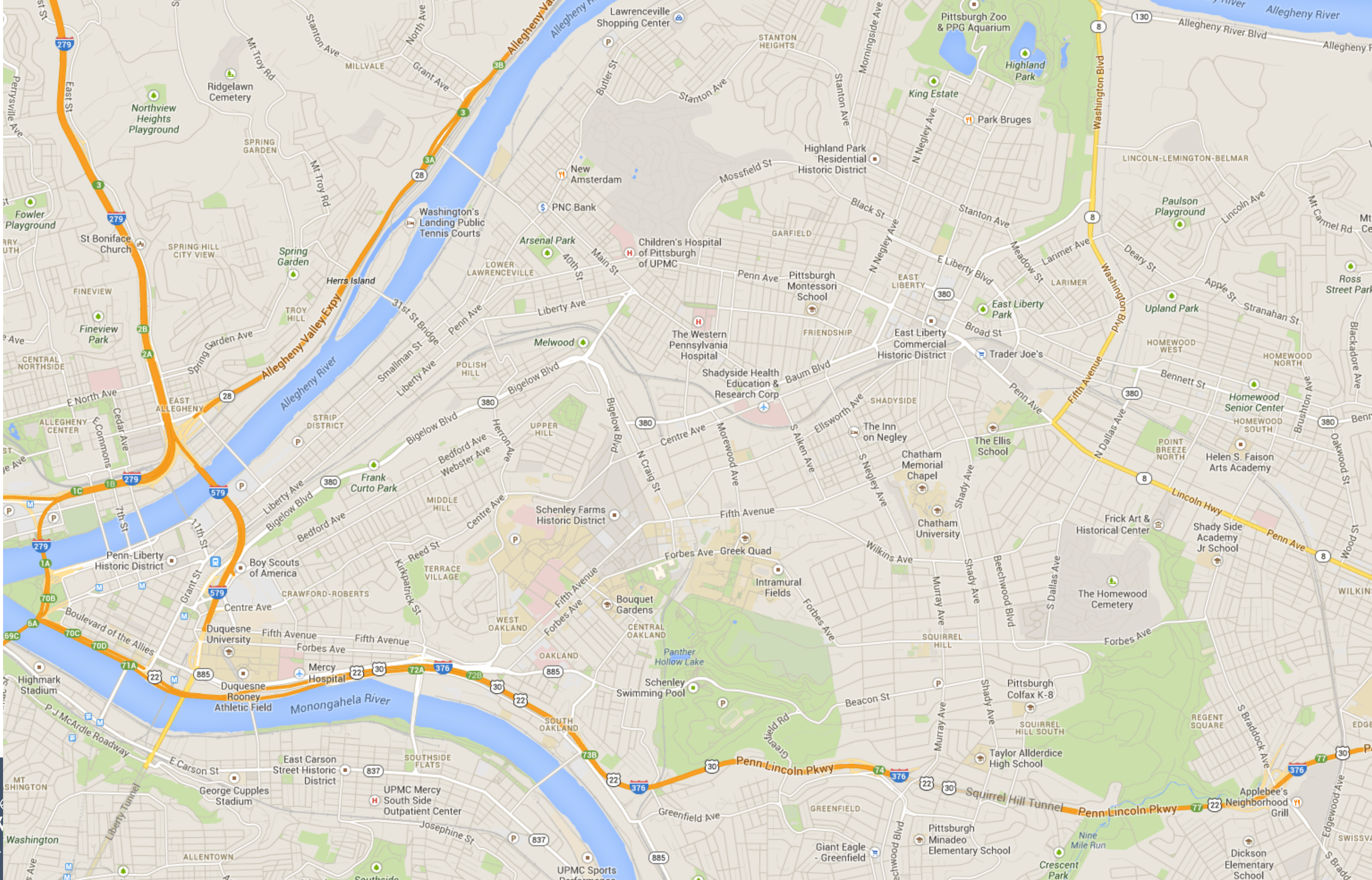
Architecture as
documentation
(representations of the system)



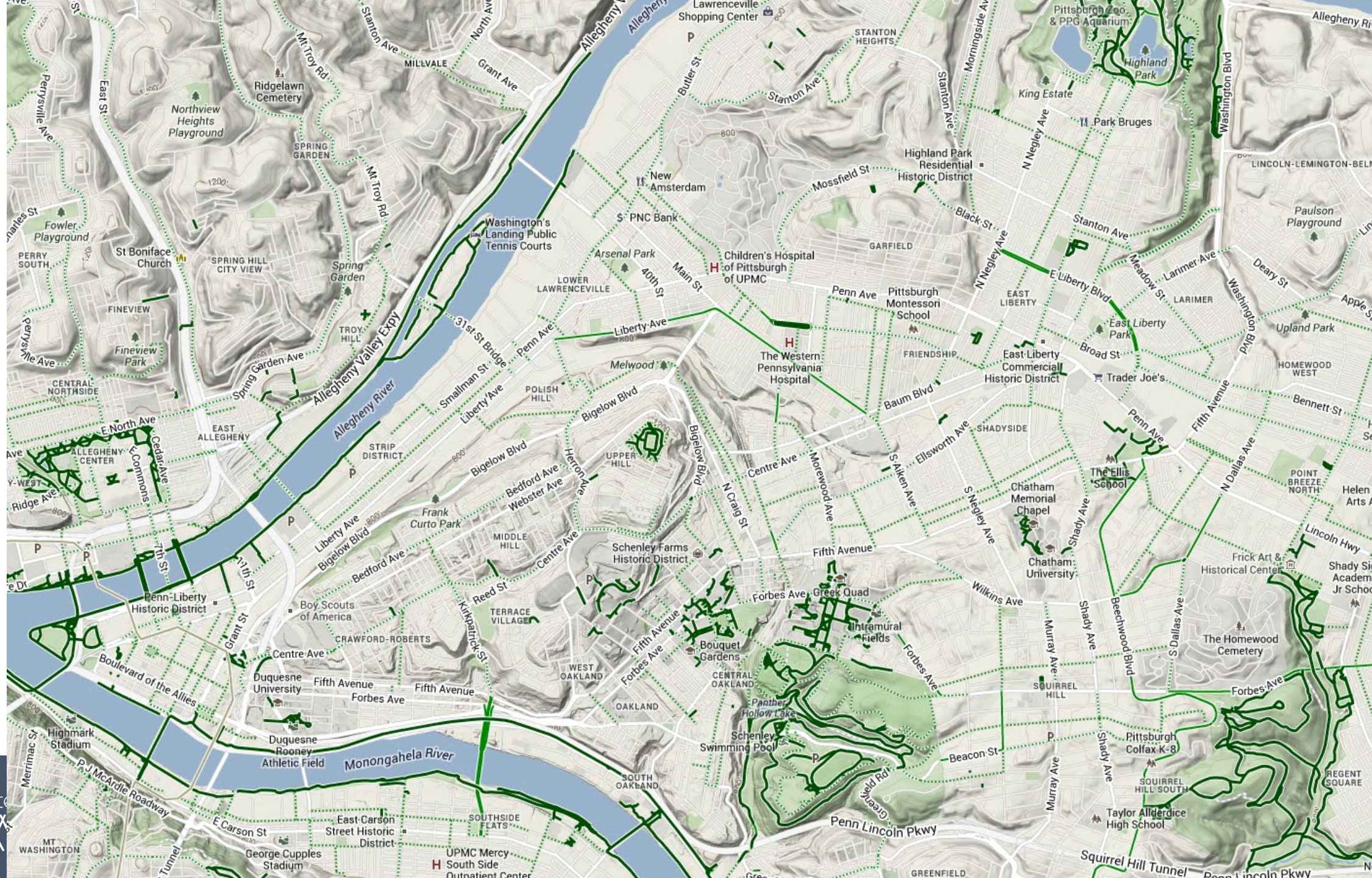
Architecture as (design) process
(activities around the other two)

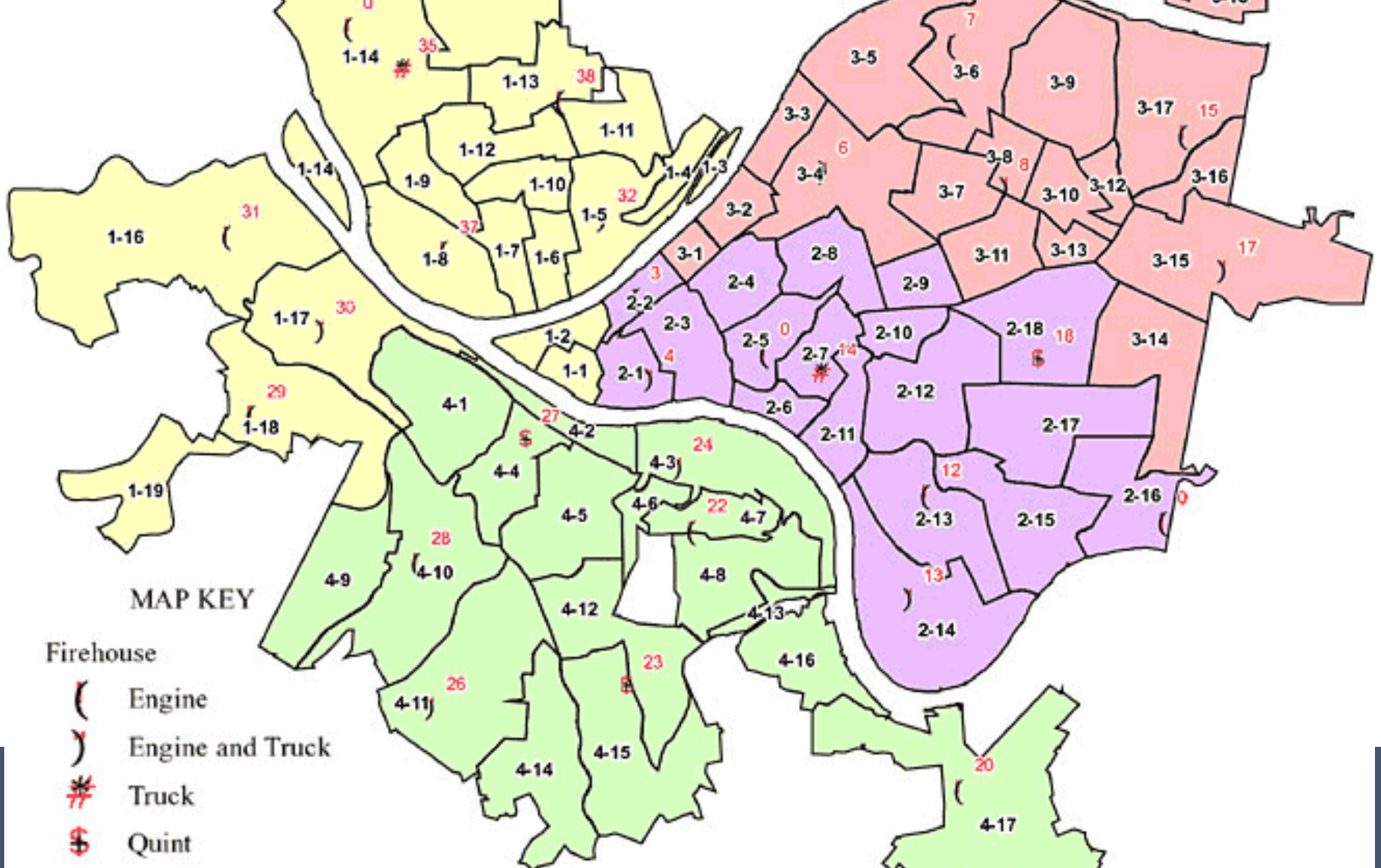
Why Document Architecture?

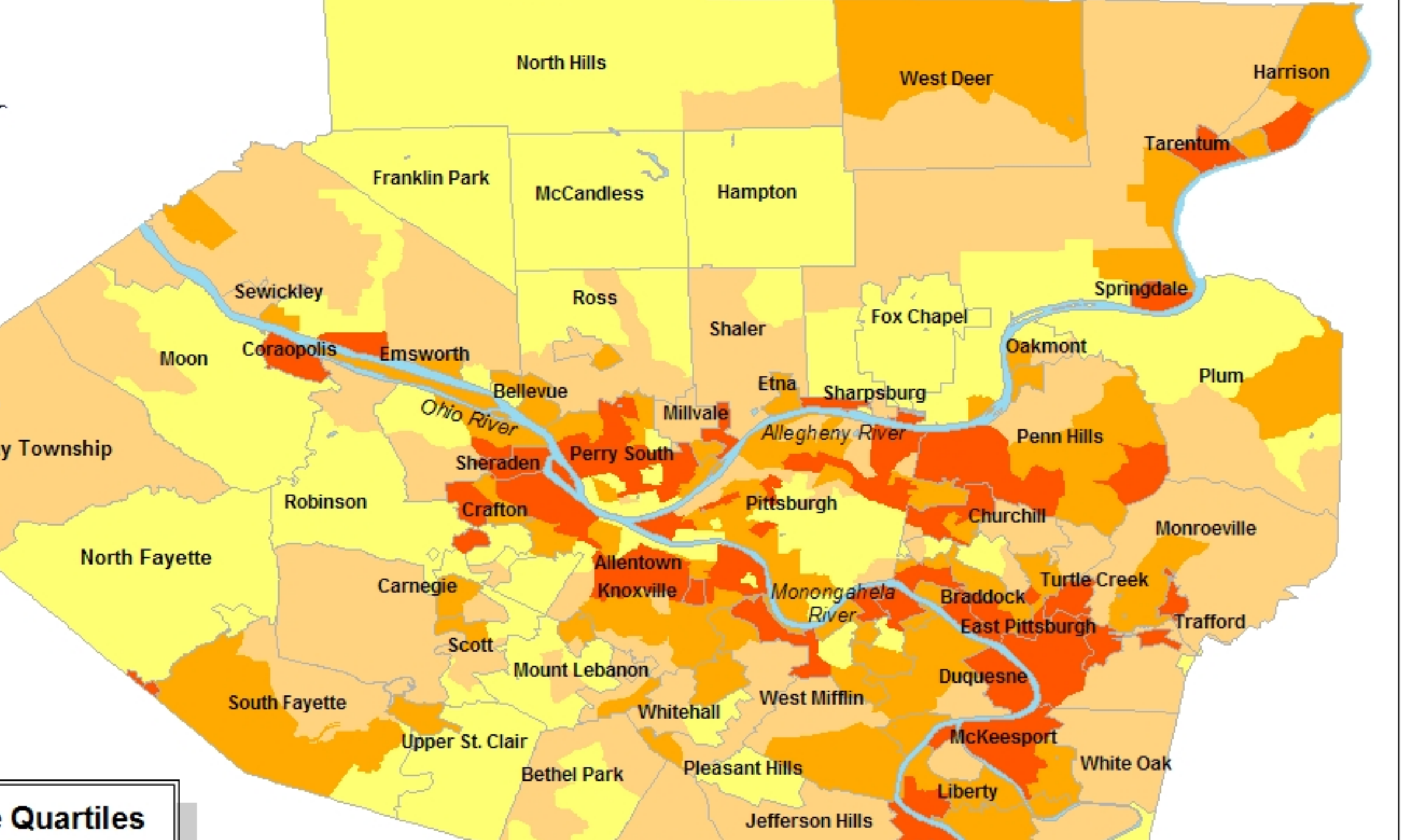
- Blueprint for the system
  - Artifact for early analysis
  - Primary carrier of quality attributes
  - Key to post-deployment maintenance and enhancement
- Documentation speaks for the architect, today and 20 years from today
  - As long as the system is built, maintained, and evolved according to its documented architecture
- Support traceability.

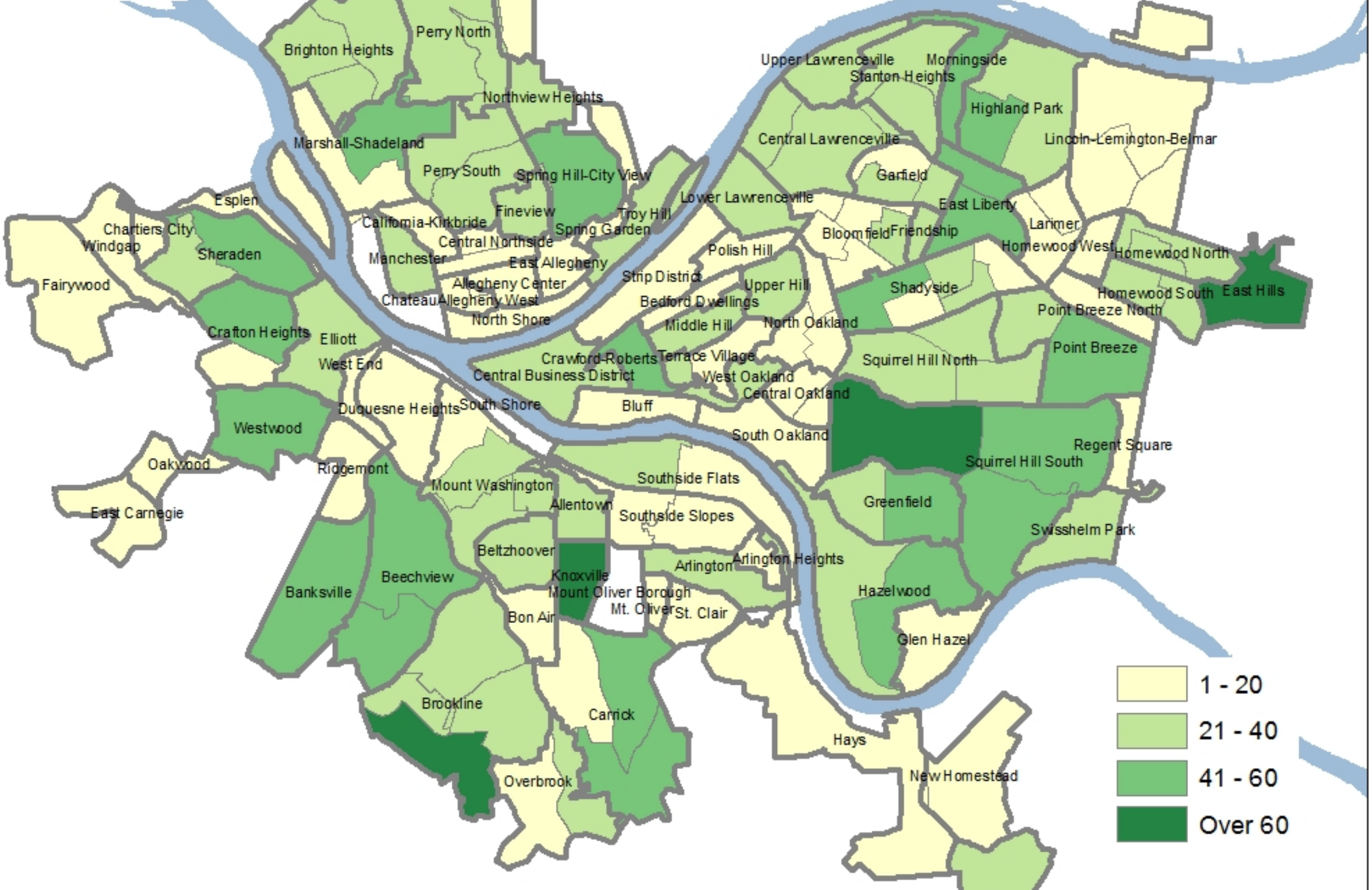


isr institute SOFTW RESEA


Allegheny River
Monongahela River
Allegheny Valley Expy
Penn Lincoln Pkwy
Squirrel Hill Tunnel
Boulevard of the Allies
Fifth Avenue
Forbes Ave
Centre Ave
Liberty Ave
Penn Ave
Bigelow Blvd
Baum Blvd
Wilkins Ave
Beacon St
Shady Ave
Ellsworth Ave
Negley Ave
Stanton Ave
Highland Park
Pittsburgh Zoo & PPG Aquarium
Lawrenceville Shopping Center
Washington's Landing Public Tennis Courts
Children's Hospital of Pittsburgh of UPMC
The Western Pennsylvania Hospital
East Liberty Commercial Historic District
Trader Joe's
Chatham University
Chatham Memorial Chapel
Schenley Farms Historic District
Schenley Swimming Pool
Bouquet Gardens
Intramural Fields
Greek Quad
Frick Art & Historical Center
The Homewood Cemetery
Pittsburgh Colfax K-8
Taylor Allderdice High School
Penn-Liberty Historic District
Duquesne University
Duquesne Rooney Athletic Field
Highmark Stadium
George Cupples Stadium
East Carson Street Historic District
UPMC Mercy South Side Outpatient Center
Ridgelawn Cemetery
Northview Heights Playground
Fineview Park
Frank Curto Park
Arsenal Park
Highland Park Residential Historic District
Pittsburgh Montessori School
East Liberty Park
Upland Park
Paulson Playground
Park Bruges
King Estate
New Amsterdam
PNC Bank
St Boniface Church
Spring Garden
Melwood
Panther Hollow Lake
Boy Scouts of America
The Ellis School
PERRY SOUTH
FINEVIEW
CENTRAL NORTHSIDE
EAST ALLEGHENY
ALLEGHENY CENTER
STRIP DISTRICT
POLISH HILL
UPPER HILL
MIDDLE HILL
TERRACE VILLAGE
CRAWFORD-ROBERTS
WEST OAKLAND
OAKLAND
CENTRAL OAKLAND
SOUTH OAKLAND
SOUTHSIDE FLATS
MT WASHINGTON
GREENFIELD
SQUIRREL HILL
SQUIRREL HILL SOUTH
REGENT SQUARE
HOMEWOOD WEST
POINT BREEZE NORTH
SHADYSIDE
FRIENDSHIP
EAST LIBERTY
LARIMER
GARFIELD
STANTON HEIGHTS
LOWER LAWRENCEVILLE
MILLVALE
SPRING GARDEN
SPRING HILL CITY VIEW
TROY HILL


MAP KEY
Firehouse
Engine
Engine and Truck
Truck
Quint


North Hills
West Deer
Harrison
Tarentum
Franklin Park
McCandless
Hampton
Springdale
Sewickley
Ross
Shaler
Fox Chapel
Oakmont
Moon
Coraopolis
Emsworth
Plum
Bellevue
Etna
Sharpsburg
Ohio River
Millvale
Allegheny River
Penn Hills
Perry South
Sheraden
Robinson
Crafton
Pittsburgh
Churchill
Monroeville
North Fayette
Allentown
Knoxville
Turtle Creek
Carnegie
Monongahela River
Braddock
Trafford
East Pittsburgh
Scott
Mount Lebanon
Duquesne
South Fayette
West Mifflin
Whitehall
McKeesport
Upper St. Clair
White Oak
Bethel Park
Pleasant Hills
Liberty
Jefferson Hills
Quartiles


Perry North
Brighton Heights
Northview Heights
Marshall-Shadeland
Perry South
Spring Hill-City View
Esplen
Fineview
Troy Hill
California-Kirkbride
Spring Garden
Chartiers City
Windgap
Central Northside
Sheraden
Manchester
East Allegheny
Fairywood
Allegheny Center
Chateau
Allegheny West
North Shore
Crafton Heights
Elliott
West End
Crawford-Roberts
Central Business District
Duquesne Heights
South Shore
Bluff
Westwood
Oakwood
Ridgemont
Mount Washington
Allentown
Southside Flats
Southside Slopes
East Carnegie
Beltzhoover
Knoxville
Arlington
Arlington Heights
Beechview
Banksville
Mount Oliver Borough
Mt. Oliver
St. Clair
Bon Air
Brookline
Carrick
Overbrook
Upper Lawrenceville
Morningside
Stanton Heights
Highland Park
Central Lawrenceville
Lincoln-Lemington-Belmar
Garfield
Lower Lawrenceville
East Liberty
Larimer
Bloomfield
Friendship
Homewood West
Homewood North
Polish Hill
Strip District
Upper Hill
Shadyside
Homewood South
East Hills
Point Breeze North
Bedford Dwellings
Middle Hill
North Oakland
Point Breeze
Terrace Village
Squirrel Hill North
West Oakland
Central Oakland
South Oakland
Regent Square
Squirrel Hill South
Greenfield
Swisshelm Park
Hazelwood
Glen Hazel
Hays
New Homestead
1 - 20
21 - 40
41 - 60
Over 60

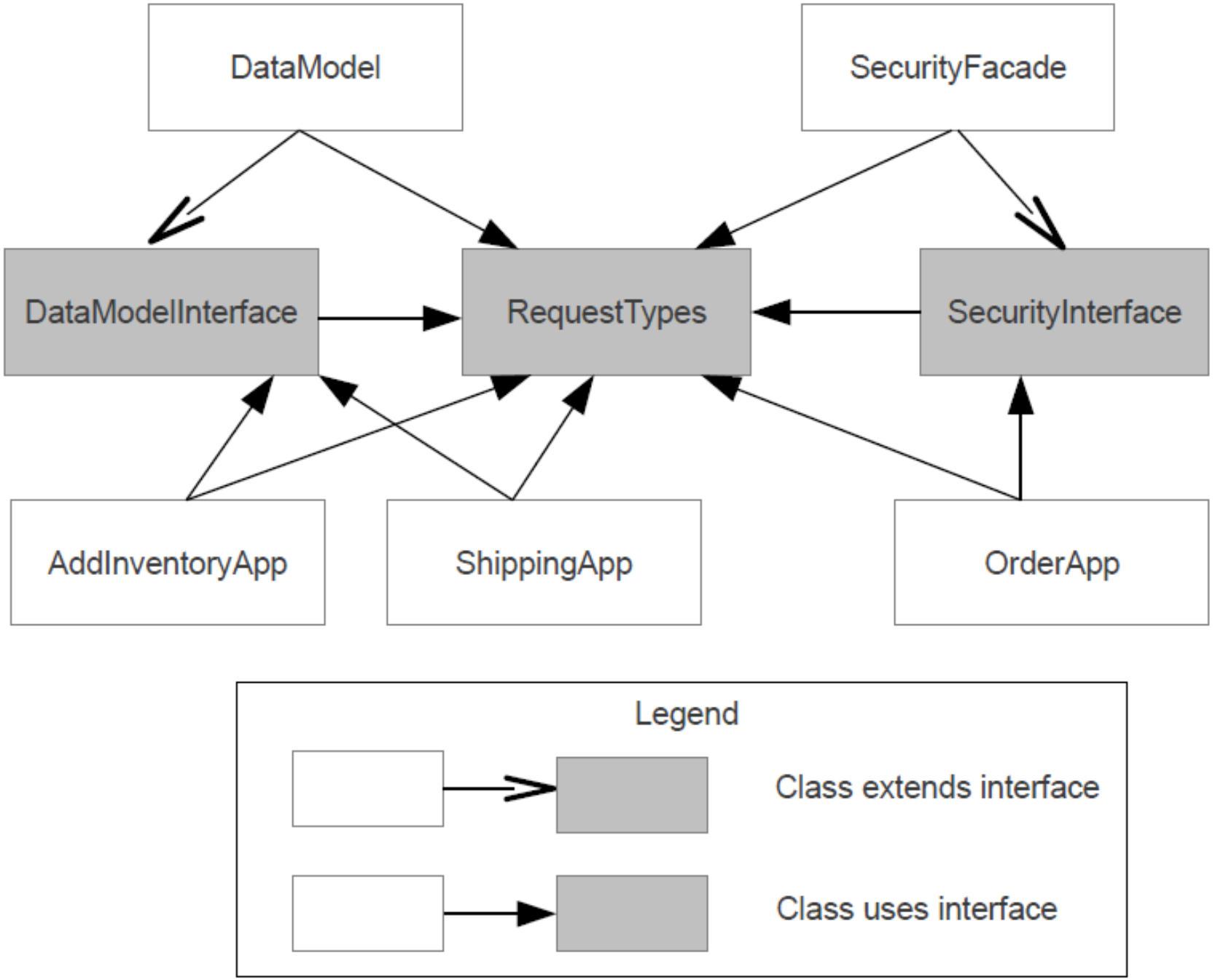

Pittsburgh
Allegheny Cemetery
Highland Park
Pittsburgh Zoo
Allegheny River
Washington Blvd
Lincoln Ave
Larimer Ave
Penn Ave
Liberty Ave
Bigelow Blvd
Centre Ave
Baum Blvd
E Busway
Ellsworth Ave
Hamilton Ave
Bennett St
Mellon Park
5th Ave
Wilkins Ave
Carnegie Mellon University
University of Pittsburgh
Homewood Cemetery
Wilkinsburg
Penn Ave
Forbes Ave
Schenley Park
Frick Park
Ninemile Run
Swissvale
Murray Ave
Hazelwood Ave
Calvary Cemetery
Johnston Ave
E Waterfront Dr
The Waterfront
E 8th Ave
Munhall
Josephine St
E Carson St
Southside Park
Becks Run Rd
Saw Mill Run Blvd
Liberty Tunnels
McKinley Park
Moore Park
Edgebrook Ave
Brookline Blvd
Grass Run
E Agnew St
Glass Run Rd
West Park
Heinz Field
Clemente Park
Point State Park
Duquesne University
2nd Ave
Spring Garden Ave
Pittview Ave
E Ohio St
River Ave
Spring Way
Smallman St
31st St
East St
Troy Rd
Butler St
Stanton Ave
Chislett St
N Negley Ave
N Euclid Ave
Lake Dr
Brighton Rd
N Charles St
Beechwood Blvd
Braddock Ave
Amity St
Ann St
279
28
579
376
22
885
51
837
8

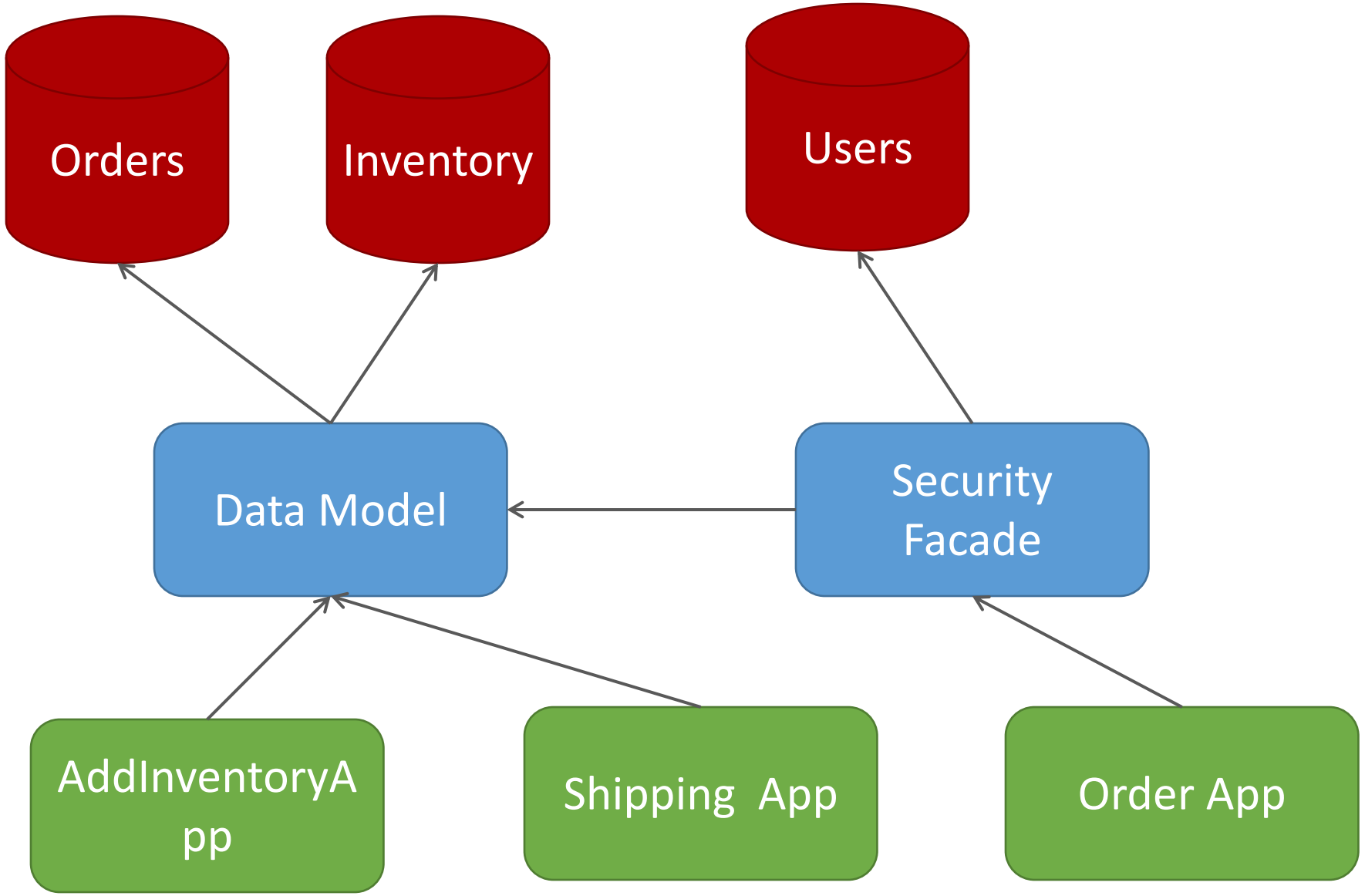
Common Views in Documenting Software Architecture

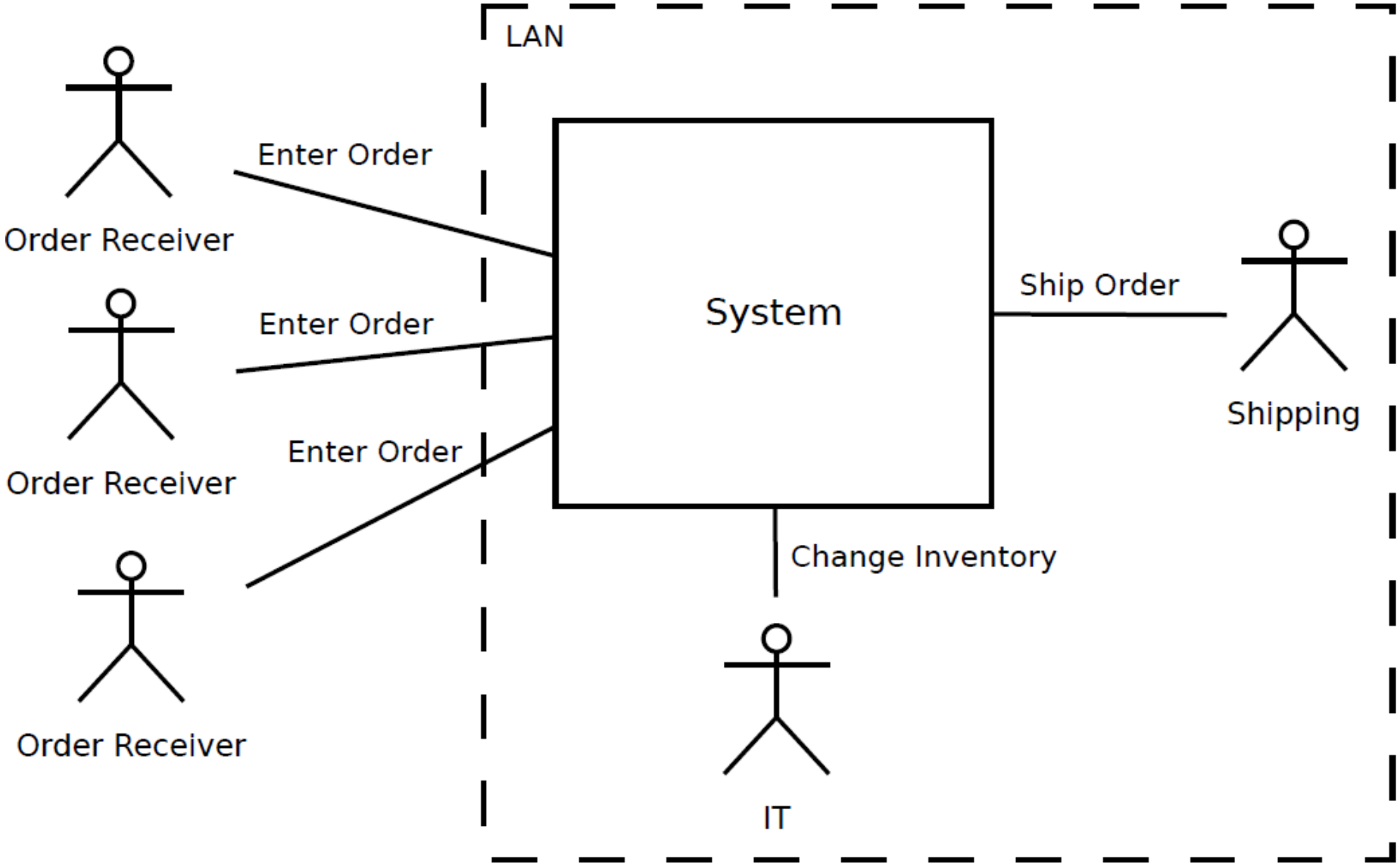
- Static View
  - Modules (subsystems, structures) and their relations (dependencies, …)
- Dynamic View
  - Components (processes, runnable entities) and connectors (messages, data flow, …)
- Physical View (Deployment)
  - Hardware structures and their connections

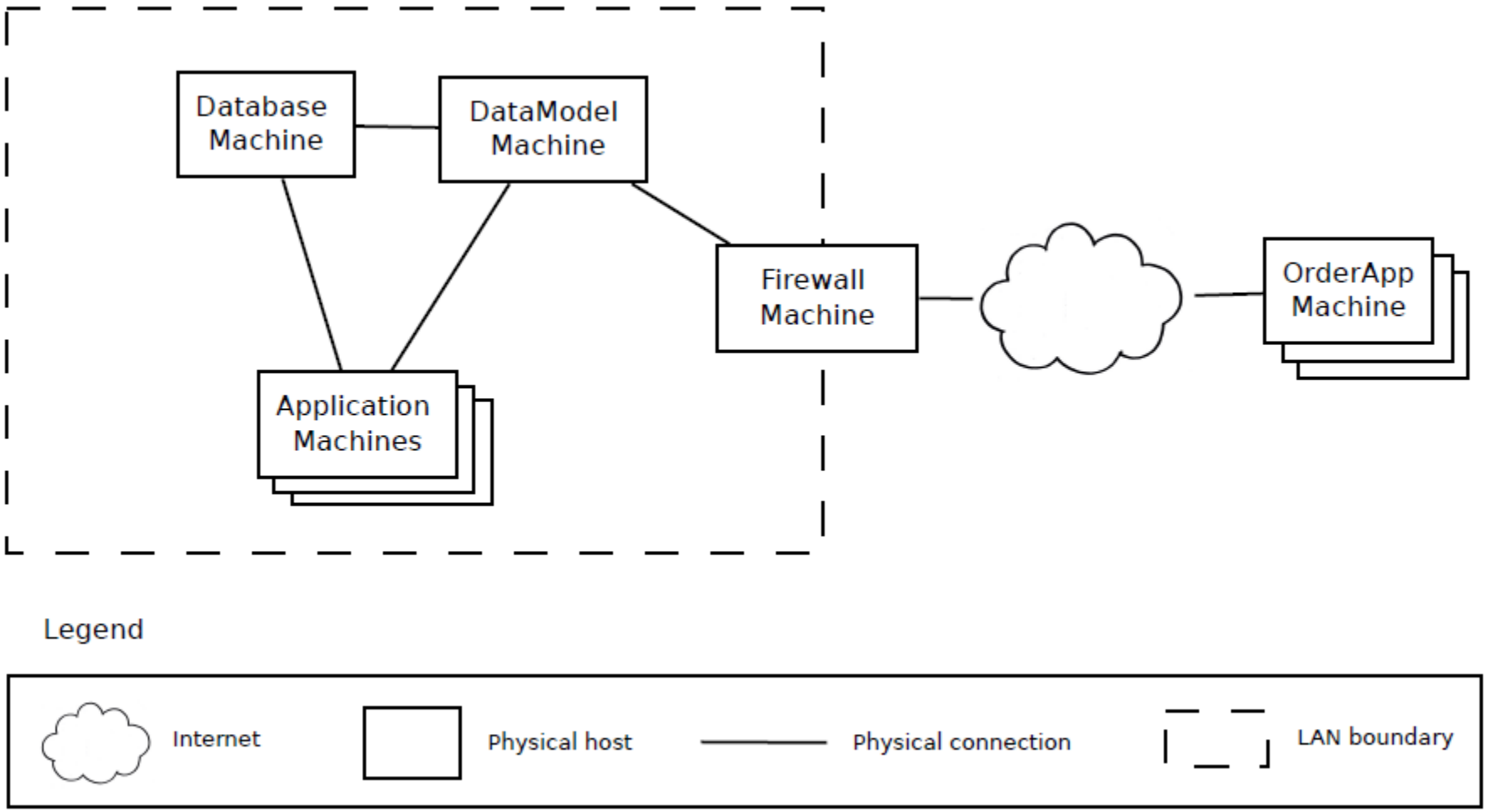
Views and Purposes

- Every view should align with a purpose
- Different views are suitable for different reasoning aspects (different quality goals), e.g.,
  - Performance
  - Extensibility
  - Security
  - Scalability
  - …


DataModel
SecurityFacade
DataModelInterface
RequestTypes
SecurityInterface
AddInventoryApp
ShippingApp
OrderApp
Legend
Class extends interface
Class uses interface


Orders
Inventory
Users
Data Model
Security Facade
AddInventoryApp
Shipping App
Order App


LAN
Enter Order
Order Receiver
Enter Order
Order Receiver
Enter Order
Order Receiver
System
Ship Order
Shipping
Change Inventory
IT


Database Machine
DataModel Machine
Firewall Machine
OrderApp Machine
Application Machines
Legend
Internet
Physical host
Physical connection
LAN boundary

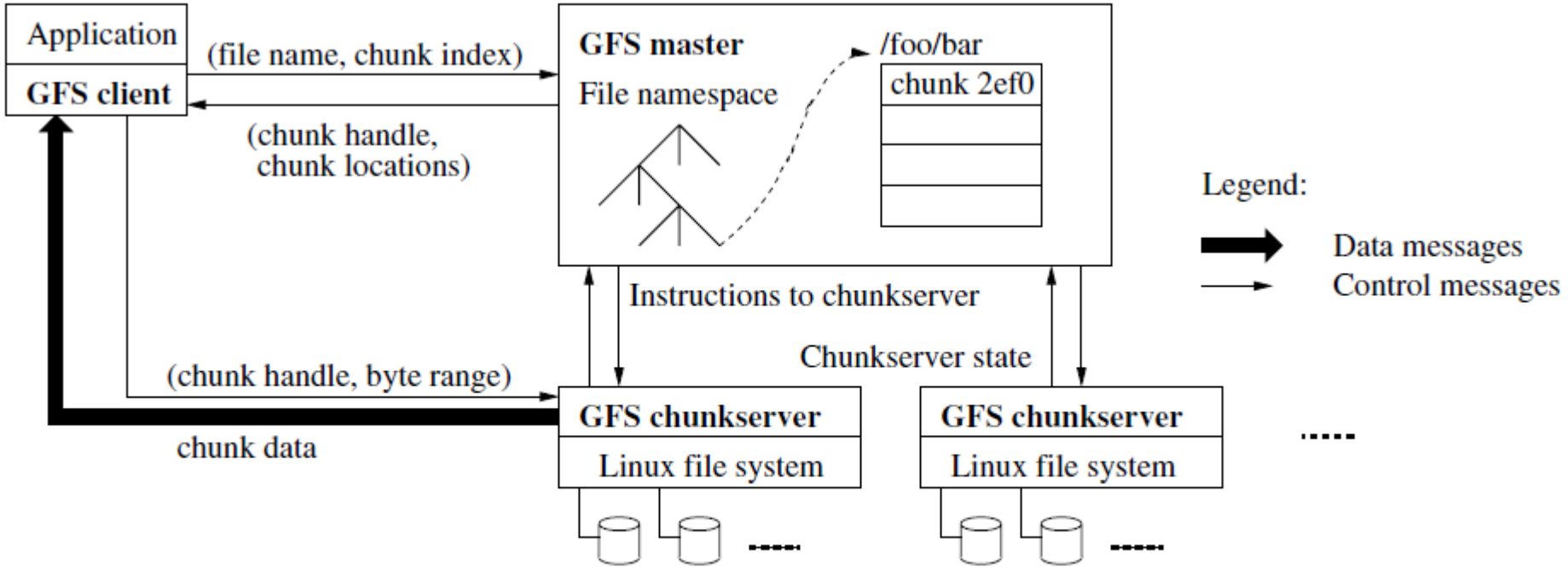


Figure 1: GFS Architecture

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS operating systems review*. Vol. 37. No. 5. ACM, 2003.

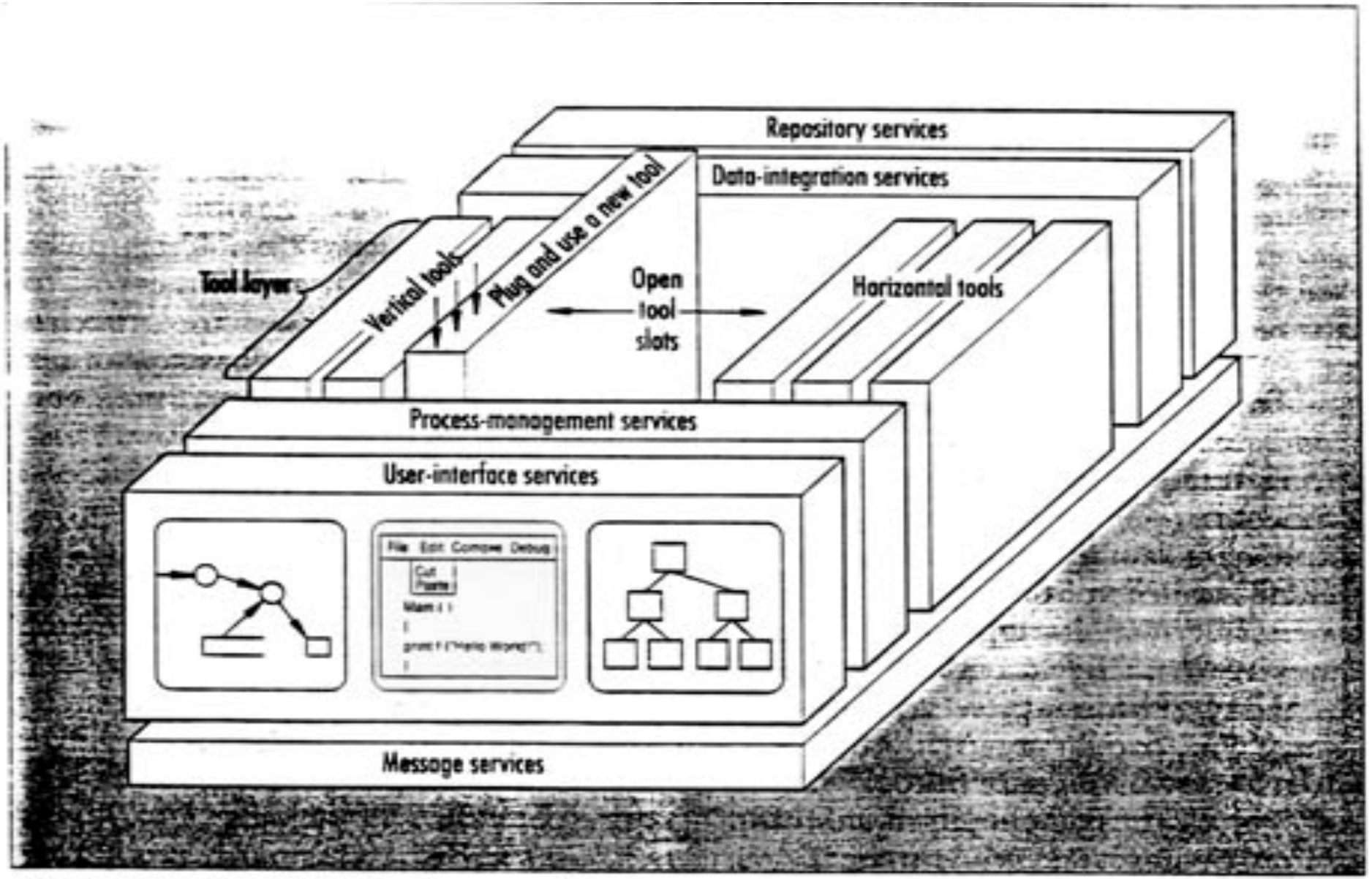


Figure 1. The NIST/ECMA reference model.

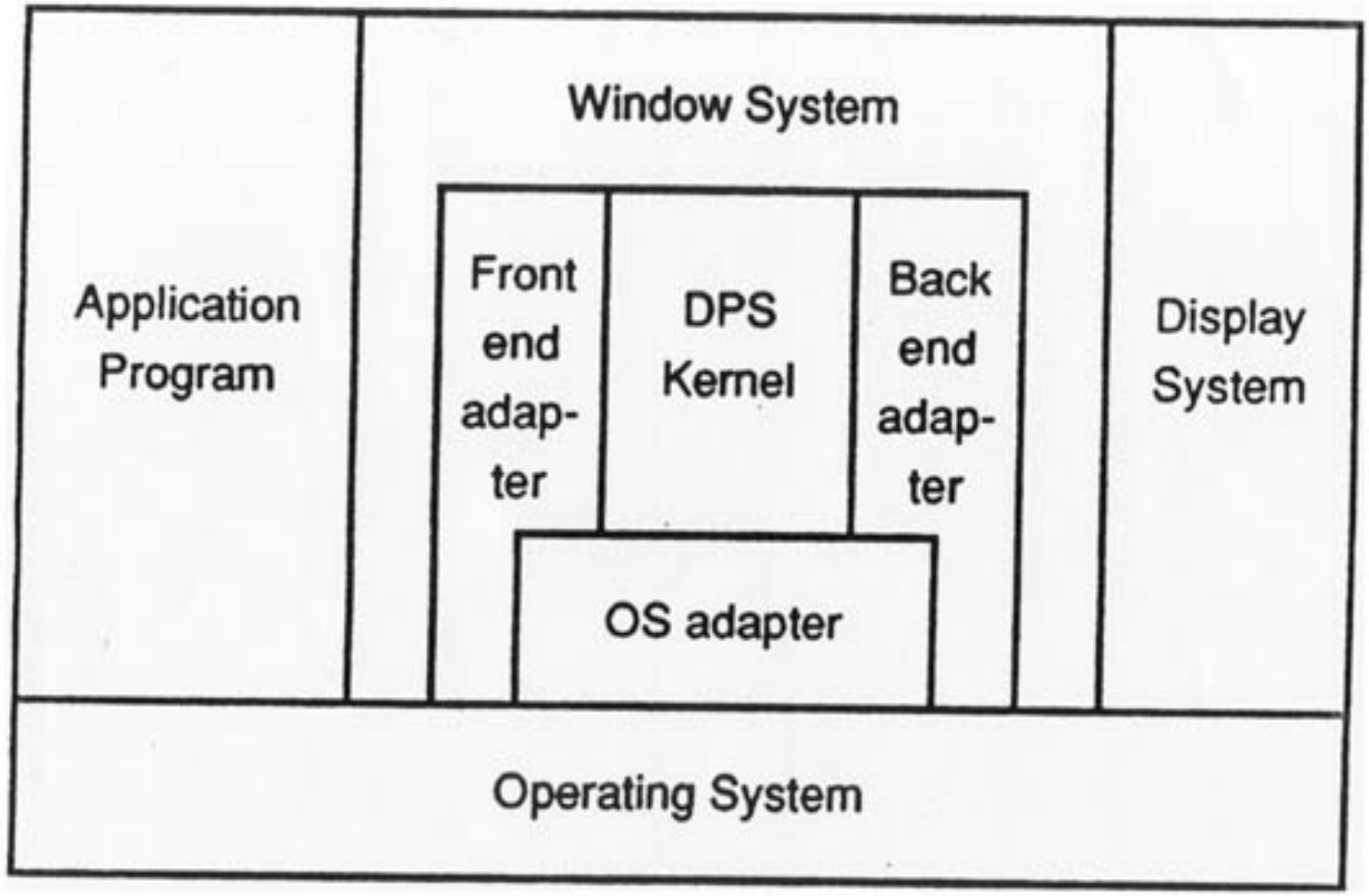


Figure 2. Display PostScript interpreter components.

An Overview of the DISPLAY POSTSCRIPT™ System, Adobe Systems Incorporated, March 16, 1988, P. 10

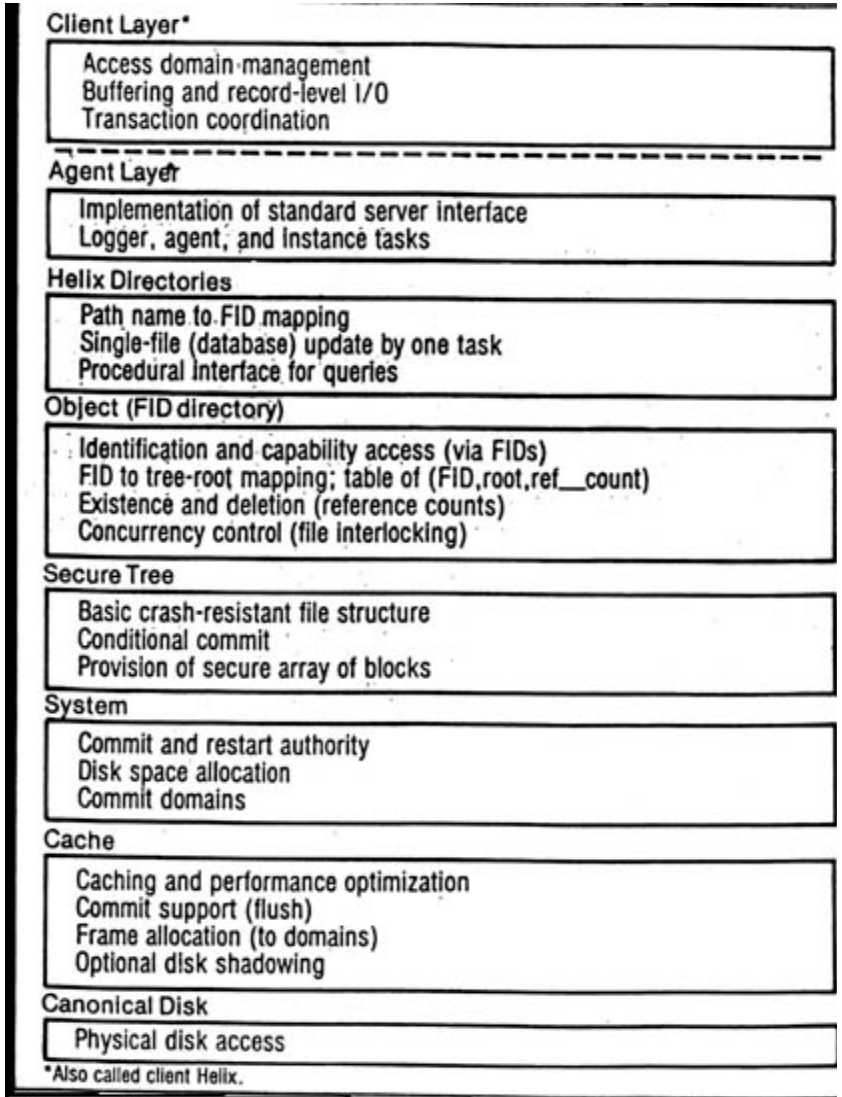


Figure 2. Abstraction layering.

IEEE Software, "Helix: The architecture of the XMS Distributed File System, Marek Fridrich and William Older, May 1985, Vol. 2, No. 3, P. 23

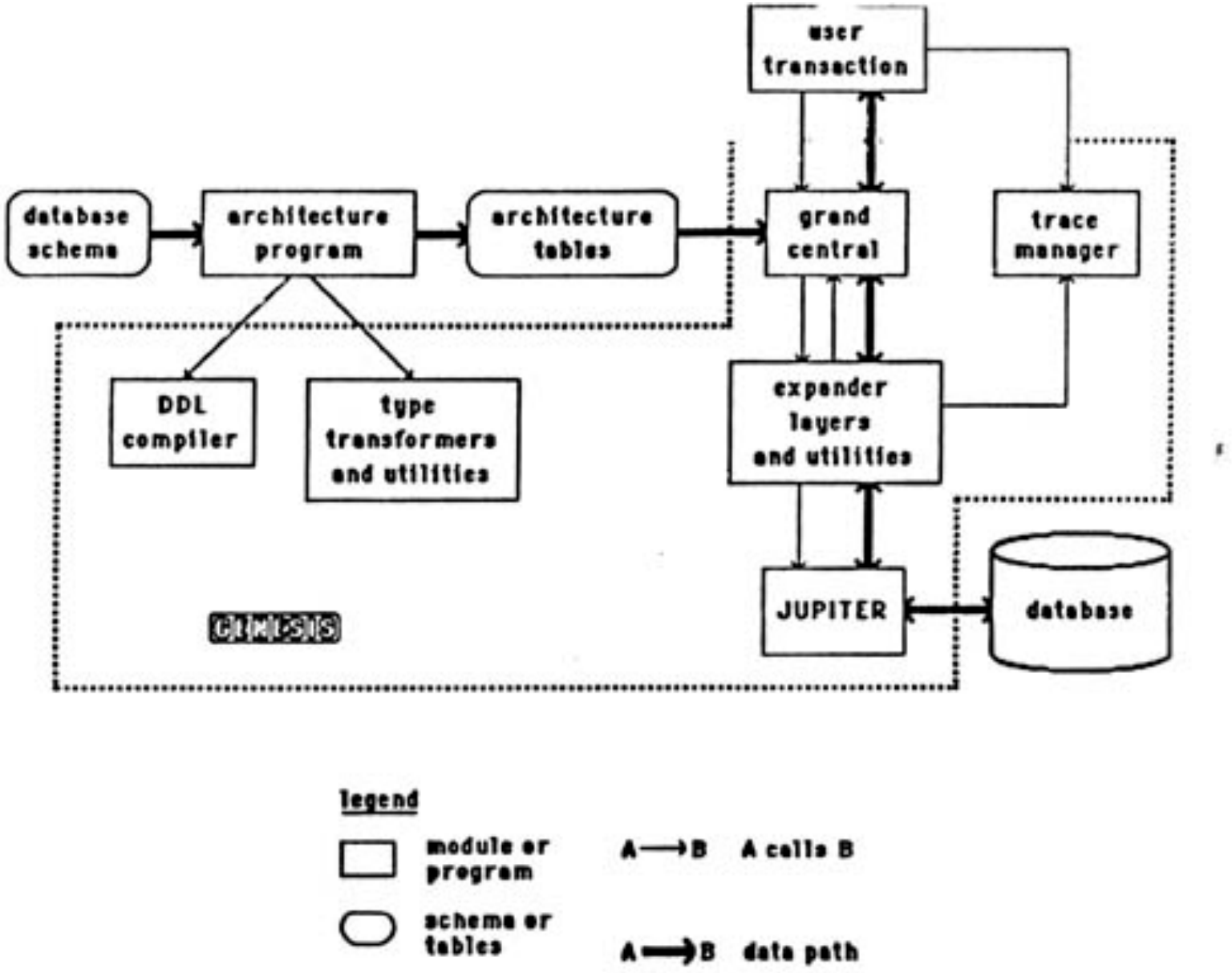
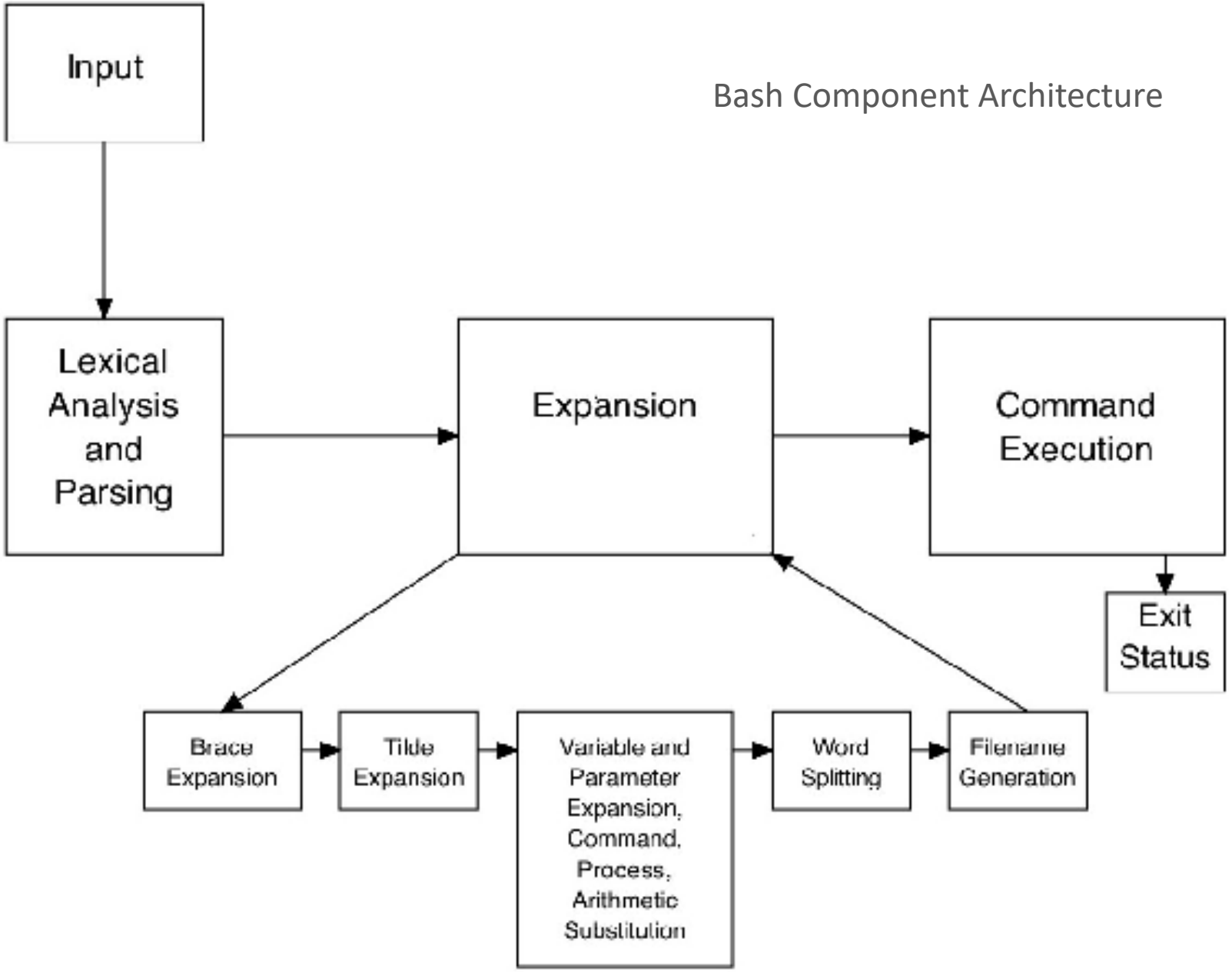


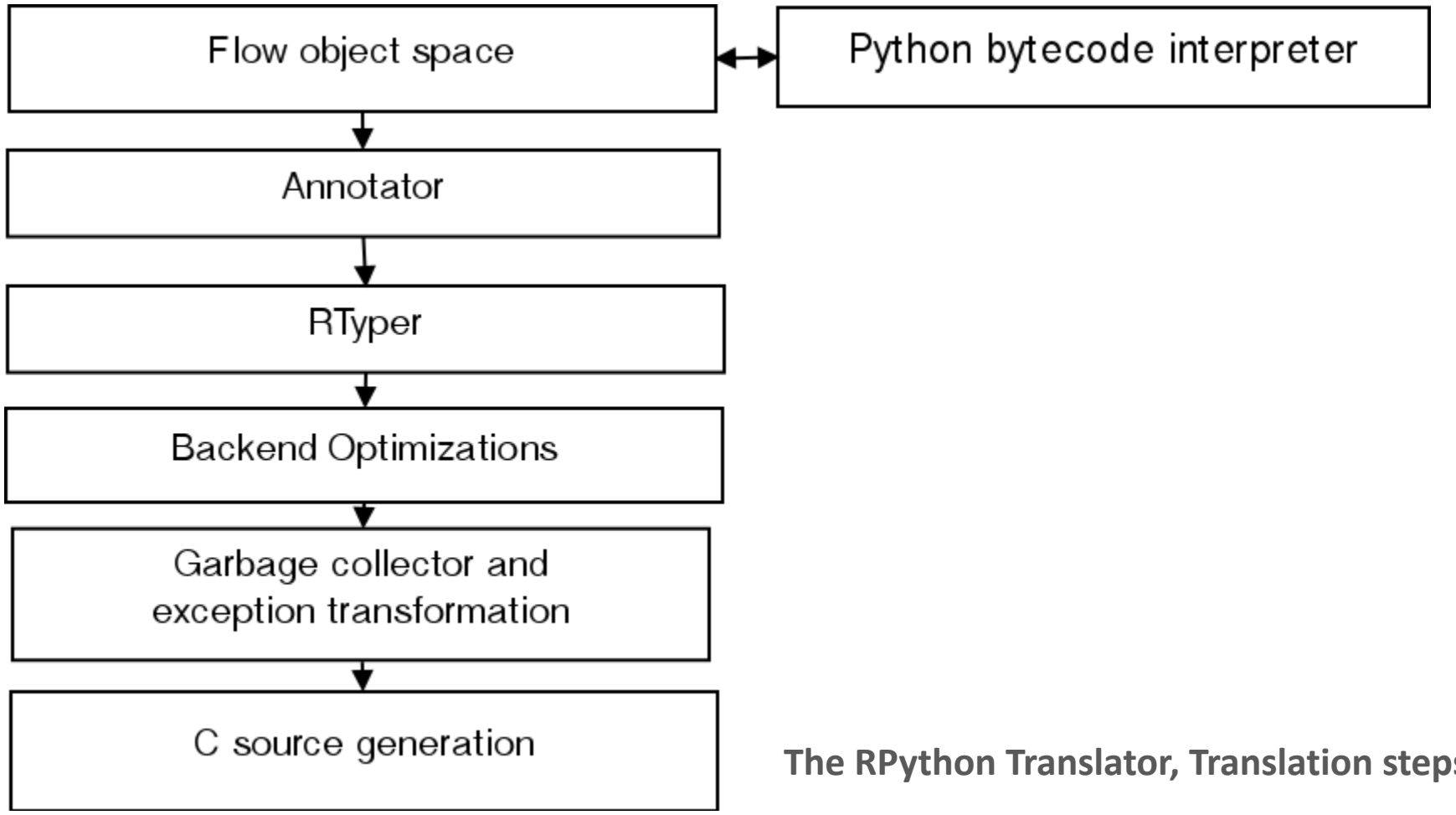
Figure 3.1 The Configuration of the GENESIS Prototype

Genesis: A Reconfiguration Database Management System, D. S. Batory, J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, Department of Computer Sciences, University of Texas at Austin,

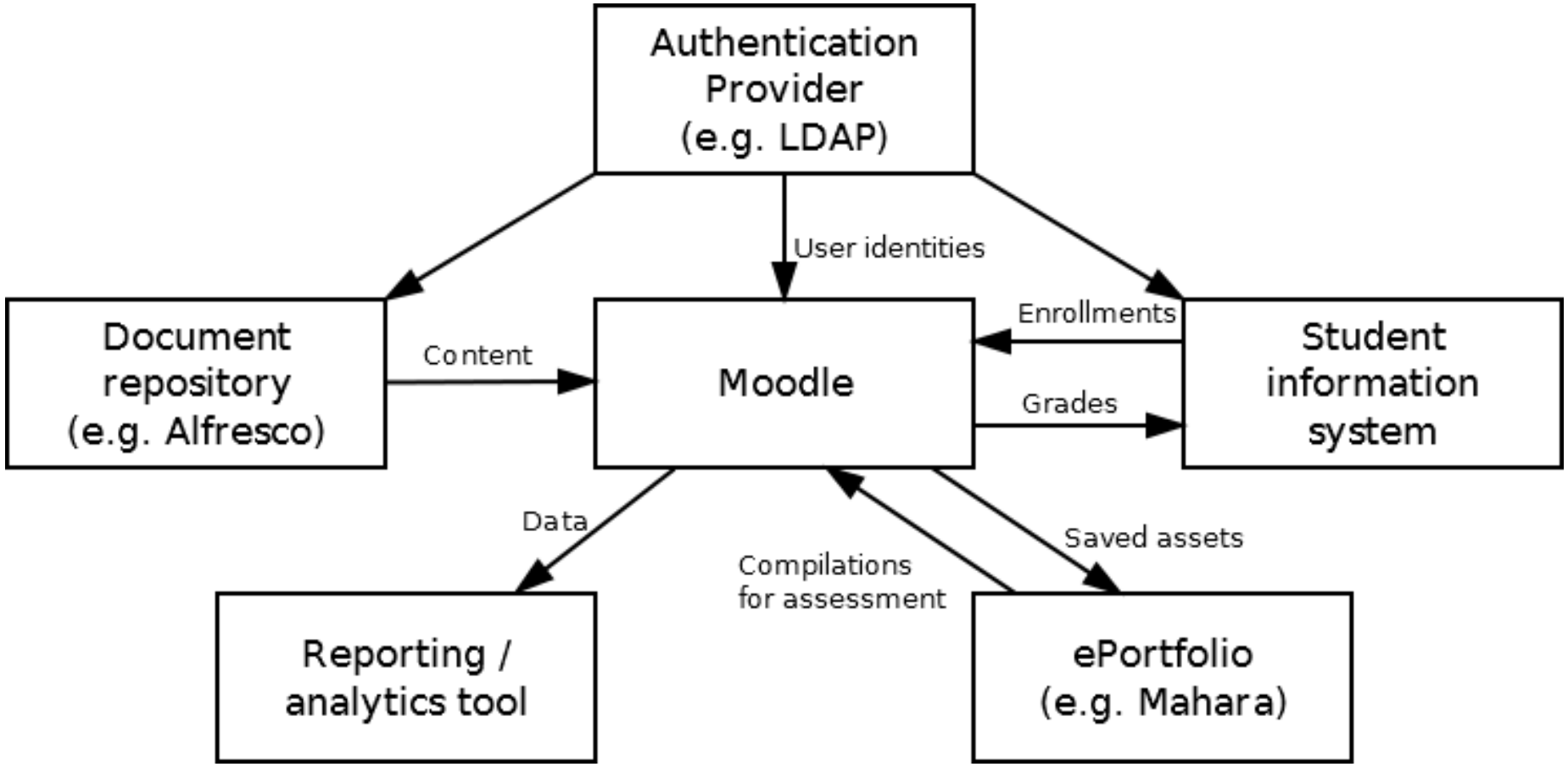
Bash Component Architecture



Example source:
http://www.aosabook.org



The RPython Translator, Translation steps



Moodle: Typical university systems architecture – Key subsystems

Selecting a Notation

- Suitable for purpose
- Often visual for compact representation
- Usually boxes and arrows
- UML possible (semi-formal), but possibly constraining
  - Note the different abstraction level – Subsystems or processes, not classes or objects
- Formal notations available
- Decompose diagrams hierarchically and in views

What is Wrong Today?

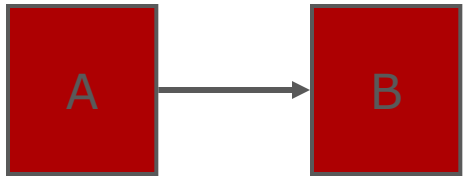
- In practice today's documentation consists of
  - Ambiguous box-and-line diagrams
  - Inconsistent use of notations
  - Confusing combinations of viewtypes
- Many things are left unspecified:
  - What kind of elements?
  - What kind of relations?
  - What do the boxes and arrows mean?
  - What is the significance of the layout?

Guidelines: Avoiding Ambiguity

- Always include a legend
- Define precisely what the boxes mean
- Define precisely what the lines mean
- Don't mix viewtypes unintentionally
  - Recall: Module (classes), C&C (components)
- Supplement graphics with explanation
  - Very important: rationale (architectural intent)
- Do not try to do too much in one diagram
  - Each view of architecture should fit on a page
  - Use hierarchy

What could the arrow mean?

- Many possibilities
  - A passes control to B
  - A passes data to B
  - A gets a value from B
  - A streams data to B
  - A sends a message to B
  - A creates B
  - A occurs before B
  - B gets its electricity from A
  - …



Recommendations for Recitation and Homework

- Use UML or UML-like notations:
  - Class diagrams for static and physical views
  - Communication diagrams for dynamic view
  - Use correct abstraction level (usually not classes/objects)
- Extend notation as needed
  - Provide a legend explaining the extensions or deviations from standard UML notation

CASE STUDY: THE GOOGLE FILE SYSTEM

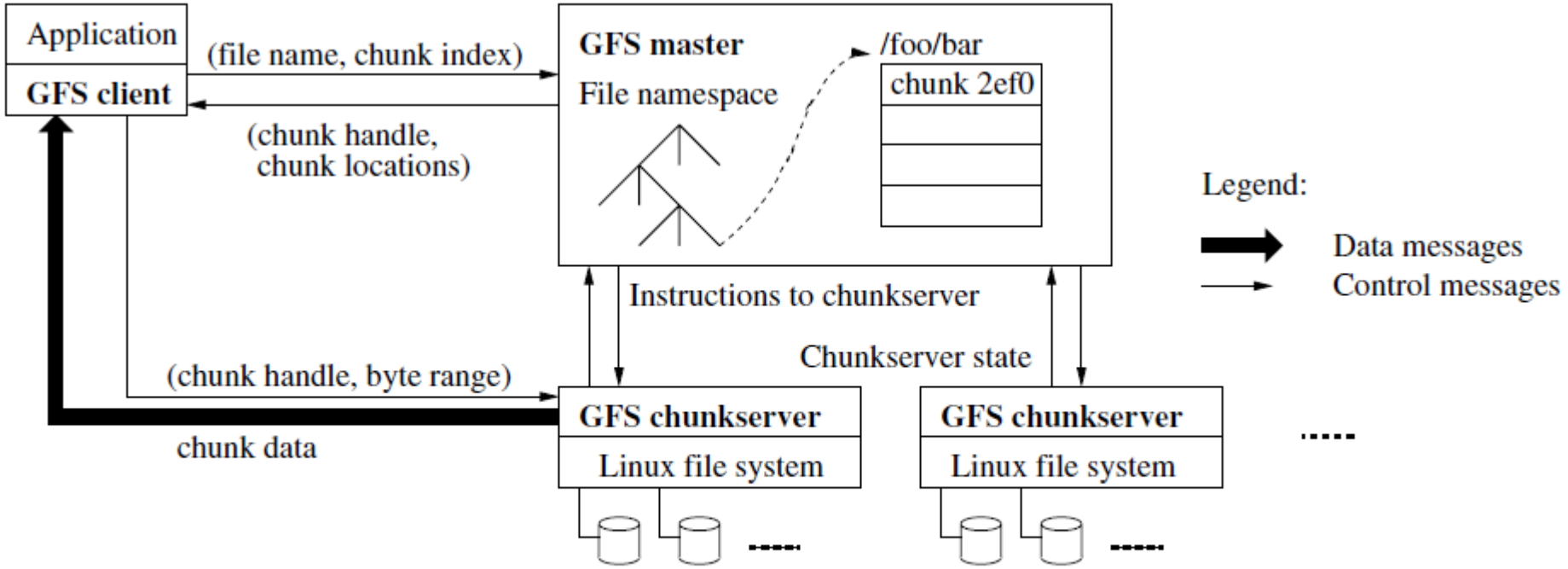


Figure 1: GFS Architecture

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS operating systems review*. Vol. 37. No. 5. ACM, 2003.

Assumptions

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS operating systems review*. Vol. 37. No. 5. ACM, 2003.

- The system is built from many inexpensive commodity components that often fail.
- The system stores a modest number of large files.
- The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.
- The workloads also have many large, sequential writes that append data to files.
- The system must efficiently implement well-defined semantics for multiple clients that concurrently append to the same file.
- High sustained bandwidth is more important than low latency.

Qualities:
Scalability
Reliability
Performance
Cost

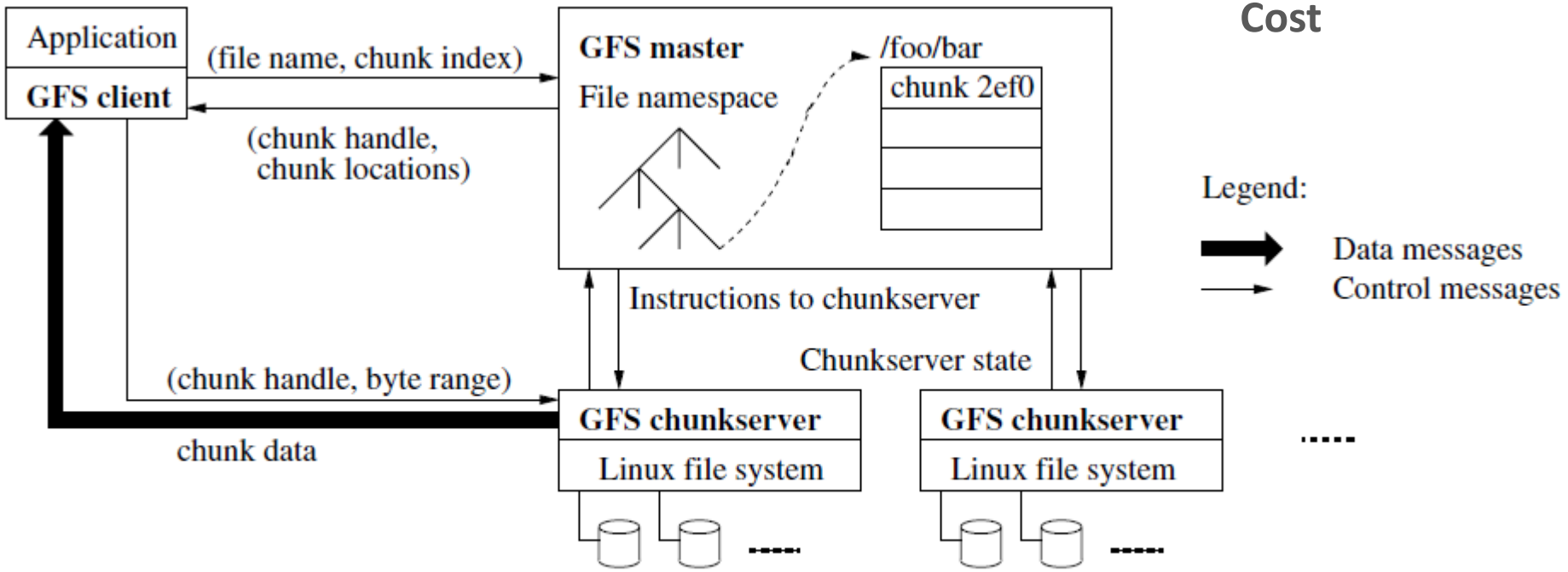


Figure 1: GFS Architecture

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS operating systems review*. Vol. 37. No. 5. ACM, 2003.

Questions

1. What are the most important quality attributes in the design?
2. How are those quality attributes realized in the design?

Qualities:
Scalability
Reliability
Performance
Cost

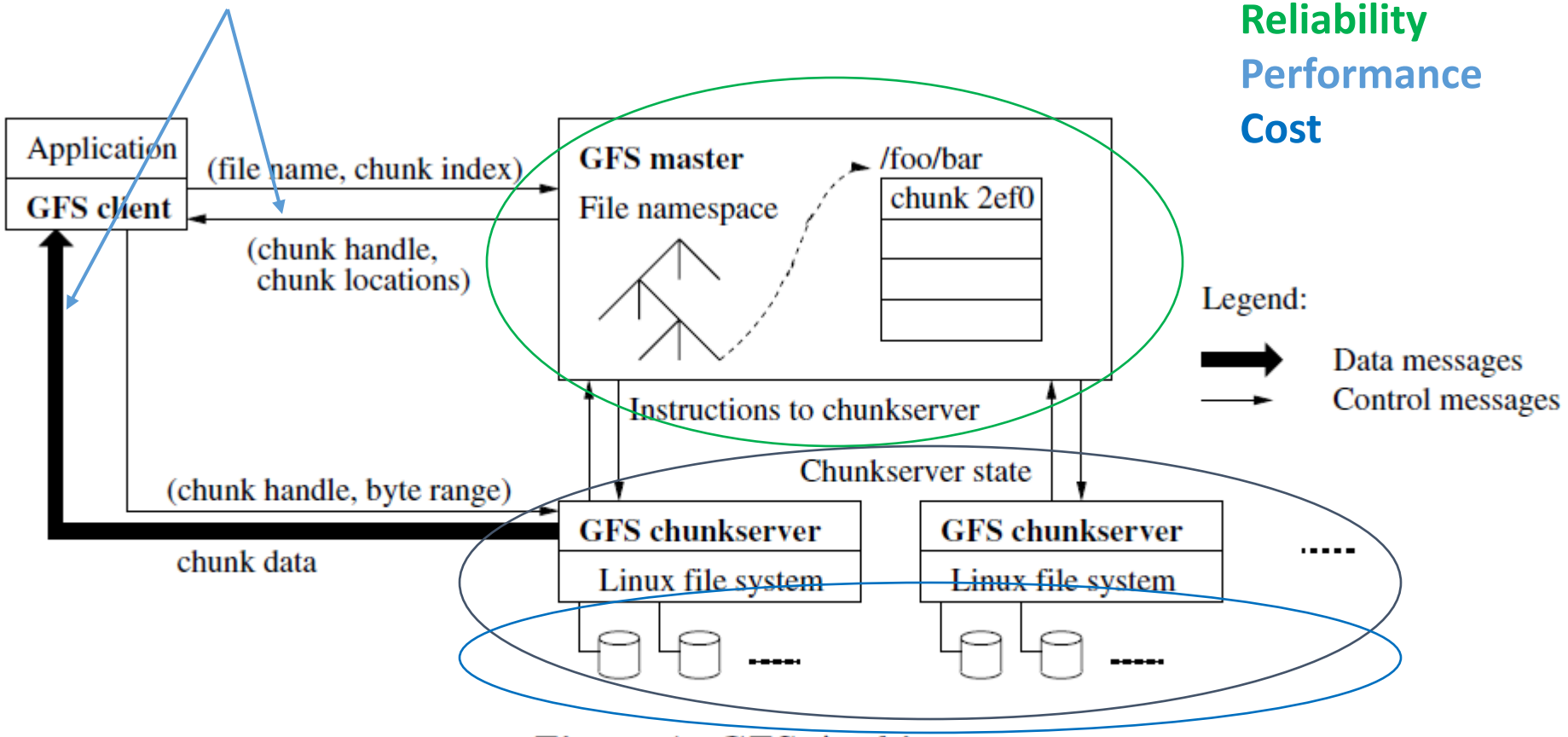


Figure 1: GFS Architecture

Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS operating systems review*. Vol. 37. No. 5. ACM, 2003.

Exercise

For the Google File System, create a physical architecture view that addresses a relevant quality attribute

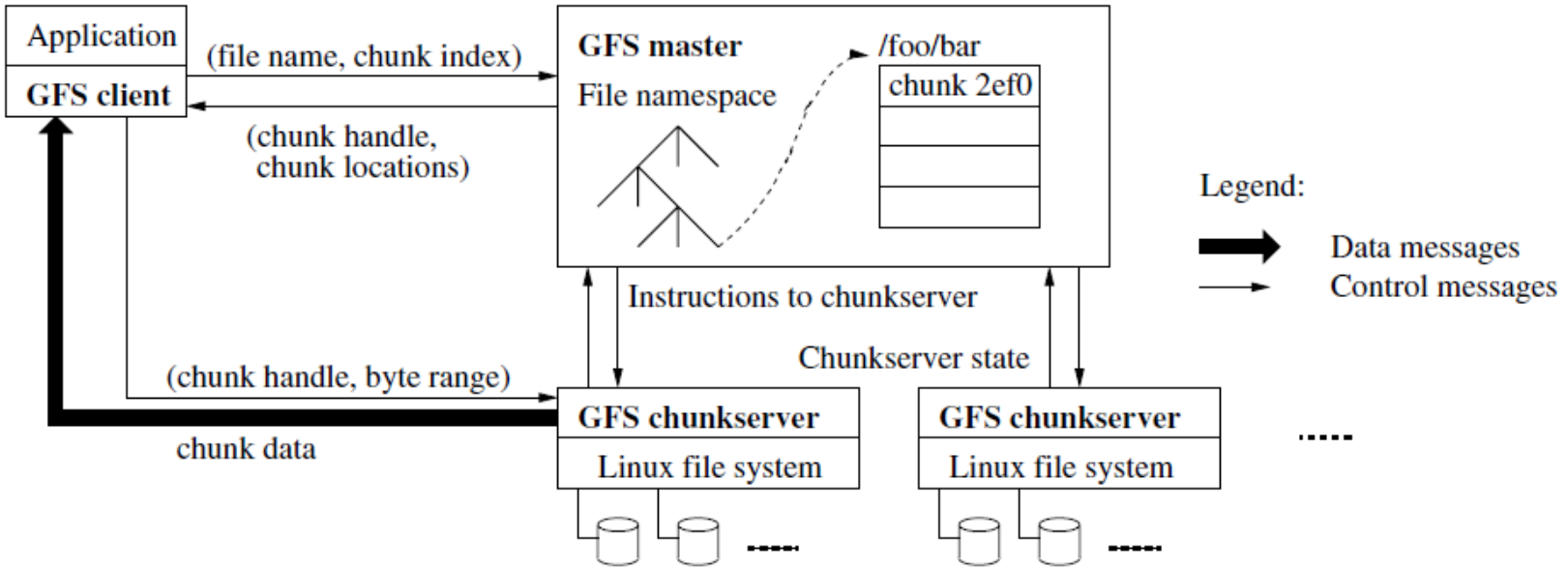


Figure 1: GFS Architecture

Further Readings

- Bass, Clements, and Kazman. Software Architecture in Practice. Addison-Wesley, 2003.
- Boehm and Turner. Balancing Agility and Discipline: A Guide for the Perplexed, 2003.
- Clements, Bachmann, Bass, Garlan, Ivers, Little, Merson, Nord, Stafford. Documenting Software Architectures: Views and Beyond, 2010.
- Fairbanks. Just Enough Software Architecture. Marshall & Brainerd, 2010.
- Jansen and Bosch. Software Architecture as a Set of Architectural Design Decisions, WICSA 2005.
- Lattanze. Architecting Software Intensive Systems: a Practitioner's Guide, 2009.
- Sommerville. Software Engineering. Edition 7/8, Chapters 11-13
- Taylor, Medvidovic, and Dashofy. Software Architecture: Foundations, Theory, and Practice. Wiley, 2009.

Further Readings

- Bass, Clements, and Kazman. Software Architecture in Practice. Addison-Wesley, 2003.
- Boehm and Turner. Balancing Agility and Discipline: A Guide for the Perplexed, 2003.
- Clements, Bachmann, Bass, Garlan, Ivers, Little, Merson, Nord, Stafford. Documenting Software Architectures: Views and Beyond, 2010.
- Fairbanks. Just Enough Software Architecture. Marshall & Brainerd, 2010.
- Jansen and Bosch. Software Architecture as a Set of Architectural Design Decisions, WICSA 2005.
- Lattanze. Architecting Software Intensive Systems: a Practitioner's Guide, 2009.
- Sommerville. Software Engineering. Edition 7/8, Chapters 11-13
- Taylor, Medvidovic, and Dashofy. Software Architecture: Foundations, Theory, and Practice. Wiley, 2009.